Constant Factor Approximation for the Weighted Partial Degree Bounded Edge Packing Problem

Pawan Aurora[1], Monalisa Jena[2] ⋆, Rajiv Raman[3]

[1] IISER, Bhopal, India
paurora@iiserb.ac.in
[2] IIIT-Delhi, India
monalisaj@iiitd.ac.in
[3] IIIT-Delhi, India and NYU, Abu Dhabi
rajiv@iiitd.ac.in

Abstract. In the partial degree bounded edge packing problem (PDBEP), the input is an undirected graph $G = (V, E)$ with capacity $c_v \in \mathbb{N}$ on each vertex v. The objective is to find a *feasible* subgraph $G' = (V, E')$ maximizing $|E'|$, where G' is said to be feasible if for each $e = \{u, v\} \in E'$, $\deg_{G'}(u) \leq c_u$ or $\deg_{G'}(v) \leq c_v$. In the weighted version of the problem, additionally each edge $e \in E$ has a weight $w(e)$ and we want to find a feasible subgraph $G' = (V, E')$ maximizing $\sum_{e \in E'} w(e)$. The problem is already NP-hard if $c_v = 1$ for all $v \in V$ [Zhang, FAW-AAIM 2012].
In this paper, we introduce a generalization of the PDBEP problem. We let the edges have weights as well as demands, and we present the first constant-factor approximation algorithms for this problem. Our results imply the first constant-factor approximation algorithm for the weighted PDBEP problem, improving the result of Aurora et al. [FAW-AAIM 2013] who presented an $O(\log n)$-approximation for the weighted case.
We also study the weighted PDBEP problem on hypergraphs and present a constant factor approximation if the maximum degree of the hypergraph is bounded above by a constant.
We study a generalization of the weighted PDBEP problem with demands where each edge additionally specifies whether it requires at least one, or both its end-points to not exceed the capacity. The objective is to pick a maximum weight subset of edges. We give a constant factor approximation for this problem.
We also present a PTAS for the weighted PDBEP problem with demands on H-minor free graphs, if the demands on the edges are bounded by polynomial. We show that the PDBEP problem is APX-hard even for bipartite graphs with $c_v = 1$, $\forall v \in V$ and having degree at most 3.

1 Introduction

Packing problems are central objects of study in the theory of algorithms. Quintessential examples of such problems are the Independent Set problem [4]

⋆ The author is supported by a TCS Scholarship in graphs, Maximum Matchings in graphs [10, 13], and the Knapsack Problem [17]. Due to their fundamental nature, and wide applicability, these problems and variants thereof have been studied intensively over several decades. In this paper, we study a variant of the matching problem that is called the Partial Degree Bounded Edge Packing problem (PDBEP for short).

In the most basic version of this problem, the input is an undirected graph $G = (V, E)$, with unit capacities on the vertices, and unit weight on the edges. The goal is to pack a maximum cardinality set $E' \subseteq E$ of edges such that in the resulting sub-graph $G' = (V, E')$, each edge $e = \{u, v\} \in E'$ is *satisfied*, where an edge is said to be satisfied if either of its end-points has degree at most 1 in the sub-graph G', i.e., $\deg_{G'}(u) \leq 1$ *or* $\deg_{G'}(v) \leq 1$.

The Maximum Matching problem, phrased as above would be to find a sub-graph $G' = (V, E')$ with maximum number of edges E' such that each edge is *satisfied*, where an edge is said to be satisfied if both of its end-points have degree at most 1 in the sub-graph G', i.e., $\deg_{G'}(u) \leq 1$ *and* $\deg_{G'}(v) \leq 1$.

The difference between the Maximum Matching problem and the PDBEP problem is only in the definition of when an edge is satisfied. While in the case of Maximum Matching, we require that the degree condition at both end-points be satisfied, we only require a weaker condition to be satisfied for the PDBEP problem, namely that for each edge, the degree condition be satisfied at at least one end-point. One can immediately observe that despite the seeming similarity with the maximum matching problem, the solutions to the two problems can be vastly different. For example, consider a star $K_{1,n}$. The maximum matching problem has a solution of size 1, whereas the PDBEP problem on the same instance has a solution of size n. In fact, while the Maximum Matching problem admits a polynomial time algorithm [10], our problem is NP-hard [20] even in the case of unit capacities.

The PDBEP problem was introduced by Zhang in [20]. He studied this problem motivated by an application in computing on binary strings [6]. The problem he studied is called the Maximum Expressive Independent Set (MEIS for short) problem where the objective is to find a subset X of maximum cardinality from a set of binary strings W such that no string $t \in X$ is *expressible* from $X \setminus \{t\}$, where a binary string s is expressible from a set of binary strings S, if it can be obtained by combining the strings in S using bitwise AND and OR operators. He studied a restricted version of this problem where each string is 2-regular which means that it has exactly two ones. This he posed as a graph problem where the graph has a vertex for every bit position and an edge $\{i, j\}$ corresponds to a string that has ones at positions i, j. Now a solution to the PDBEP problem with uniform $c_u = 2$ corresponds to a subset of strings such that for any string in the subset with ones at positions i, j, at most one other string can have a 1 at one of these two positions which means that the subset of edges gives a solution to the MEIS problem (this follows from Lemma 2.4 in [20]).

Another natural application of the PDBEP problem is in resource allocation. Here, we are given $|V|$ types of resources and $|E|$ jobs. Each job needs two types of resources. A job u can be accomplished if either one of its necessary resources is shared by no more than c_u other jobs. The problem then asks to finish as many jobs as possible. In many natural settings a job may be allocated a larger set of resources than is actually required, or alternately, the job may still be able to complete with a subset of the resources allocated. A natural setting is for example, allocating man-power for projects. Some project might be allocated with slightly larger teams than is essential to complete, but might be satisfied with a possibly smaller subset. The PDBEP problem models these scenarios.

We now formally define the problems studied in this paper. In the PDBEP problem, the input is an undirected (multi-)graph $G = (V, E)$ with a capacity $c_v \in \mathbb{N}$ on each vertex v. We want to compute a subgraph $G' = (V, E')$ of G with maximum $|E'|$ such that each edge in E' is *satisfied*. We say that an edge is satisfied if at least one of its end-points is not overloaded. Thus, we want for each edge $e = \{u, v\} \in E'$, $|\delta'(u)| \leq c_u$ or $|\delta'(v)| \leq c_v$, where for any vertex v_i, $\delta'(v_i)$ denotes the set of edges in E' incident on v_i. For an edge $e = \{u, v\}$, an end-point that is not overloaded, we call a *good end-point* of e.

In the weighted PDBEP problem, each edge e is additionally associated with a weight w_e and the objective is to find a subgraph $G' = (V, E')$ such that $w(E')$ is maximized and each edge $e \in E'$ is satisfied, where $w(E') = \sum_{e \in E'} w(e)$. We also study the unit-capacity case, i.e., $c_v = 1$, $\forall v \in V$. We use PEp to denote the weighted PDBEP problem with unit-capacity.

We also consider the weighted PDBEP Problem with Demands, denoted PEpD and defined as follows: Given an undirected (multi-)graph, with $w, d : E \to \mathbb{N}$, the weights and demands respectively, on the edges, and $c : V \to \mathbb{N}$ the capacity on the vertices. We want to find a sub-graph $G' = (V, E')$ such that $w(E')$ is maximized, and each edge in E' is satisfied, where an edge $e = \{u, v\}$ is said to be satisfied, if $d(\delta'(u)) \leq c_u$ or $d(\delta'(v)) \leq c_v$, where for any $F \subseteq E$, $d(F) = \sum_{e \in F} d_e$.

The rest of the paper is organized as follows. In Section 2, we present the notation used, and present preliminary results. Section 3 describes related work. We study the PEp problem in Section 4, and then present results for the PEpD problem in Section 5. In Section 6, we study a generalized version of the PEpD problem and give a constant factor approximation algorithm for it. In Section 7, we study the weighted PDBEP problem on hypergraphs of constant degree. We give a PTAS for the weighted PDBEP problem on H-minor free graphs in Section 8. In Section 9, we prove that the PDBEP problem is APX-hard.

2 Preliminaries

Let $G = (V, E)$ denote an undirected graph. In our setting, the graphs come equipped with a weight function $w : E \to \mathbb{N}$ and a demand function $d : E \to \mathbb{N}$ on the edges, and a capacity function $c : V \to \mathbb{N}$ on the vertices. We also consider the special case when all vertices have unit capacity. In this setting, we assume that the demand of each edge is also 1, and that the graphs are simple. When we consider the general problem, the graph is no longer assumed to be simple.

We also consider directed graphs, denoted $D = (V, A)$. Each edge $(u, v) \in A$ with head v and tail u is said to be *entering* v, and *exiting* u. We use $in(v), out(v)$ respectively to denote the edges entering and exiting v.

For a vertex v, we let $\delta(v) = \{e = \{u, v\} \in E\}$ denote the set of edges incident on v. In the weighted setting, we let $\delta_{\max}(v) = \{e \in \delta(v) : w(e) \geq w(f)\ \forall f \in \delta(v)\}$. Thus, $\delta_{\max}(v)$ is the set of heaviest weighted edges incident on v. For a set $F \subseteq E$ of edges, we let $w(F) = \sum_{e \in F} w(e)$, and $d(F) = \sum_{e \in F} d_e$.

We also consider in this paper, two graph orientation problems. The Maximum Degree-Bounded Orientation Problem with Demands (ORD) is defined as follows: The input is identical to the PEPD problem, namely an undirected graph $G = (V, E)$, $w, d : E \to \mathbb{N}$ and $c : V \to \mathbb{N}$. The goal is to select a maximum weight subset of edges $E' \subseteq E$, and compute an orientation $\overrightarrow{E'}$ of the edges in E' such that the total demand of $in(v)$ is at most its capacity for each $v \in V$, i.e, $\sum_{e \in in(v)} d_e \leq c_v$ for all $v \in V$. When all vertex capacities are 1, we assume $d_e = 1,\ \forall e \in E$, and use OR to denote this problem with unit capacity and demands.

A solution to PEPD yields a solution to the ORD problem on the same instance. To see this, each edge e in a feasible PEPD solution has a good end-point, and orienting e towards its good end-point is a feasible solution to ORD of the same weight. It would be tempting to hope that the reverse might be true; and if so, this would be cause for cheer as we will show that the ORD problem is tractable. Unfortunately, this is not the case even in the unit-capacity case. Consider for example, a triangle with unit capacity on the vertices, and unit weight on the edges. Any feasible solution to the PEP problem consists of at most 2 edges, but orienting the three edges in a cycle is feasible for OR. In fact, it is known that the PEP problem is NP-hard [20].

Our approximation algorithms for the PEPD problem nevertheless use a solution to the ORD problem as a starting point, and in fact, any solution to the ORD problem can be transformed into one for the PEPD problem on the same instance at the loss of a small constant factor. The relation between the two problems is useful, and we capture this in the following proposition.

Proposition 1. *For any instance I, $OPT_{\text{PEPD}}(I) \leq OPT_{\text{ORD}}(I)$.*

We show that OR problem can be reduced in polynomial time to the b-matching problem in bipartite graphs, which can be solved in polynomial time[18]. Hence, OR can be solved in polynomial time. Similarly, the ORD problem with demands can be reduced to the demand matching problem in bipartite graphs [19], and hence, 2-approximation follows.

Lemma 1. *The* OR *problem can be solved in polynomial time.*

Proof. We prove this by giving a polynomial time reduction from OR to b-matching in bipartite graphs. The reduction is as follows: Given an instance $G = (V, E)$, $c : V \to \mathbb{N}$, $w : E \to \mathbb{N}$ of the OR problem, we construct a bipartite graph $H = (E \cup V, F)$, with capacities $b_e = 1$ for all $e \in E$ and $b_v = c_v$ for all

$v \in V$. The edges F are defined as follows: For each edge $e = \{u, v\}$, add two edges $\{e, u\}, \{e, v\}$ to F, each of weight $w(e)$.

Suppose $E' \subseteq E$ is a feasible solution to Or, where $\overrightarrow{E'}$ denotes a feasible orientation of E'. We claim that E' yields a feasible matching M in H of the same weight. Corresponding to each $e = (u, v) \in \overrightarrow{E'}$, pick $\{e, v\}$ in M. Then, exactly one edge is chosen in M for each $e \in E'$, and for each v, at most c_v edges in M are incident on it. Thus, M is feasible and has weight $w(M) = w(E')$.

Let M be a maximum weight b-matching in H. Let $E' \subseteq E$ be the set of edges of G covered by M. We claim that E' is a feasible solution to Or. To see this, since $b_e = 1$ for all $e \in E$, for each $e = \{u, v\}$, at most one of $\{e, u\}$ or $\{e, v\}$ is in M. This defines an orientation of e in the graph G. If $\{e, u\} \in M$, let $e \in E'$ and $\overrightarrow{e} = (v, u)$. Else, if $\{e, v\} \in M$, let $e \in E'$ and let $\overrightarrow{e} = (u, v)$. Since edges $\{e, v\}$ and $\{e, u\}$ have the same weight as that of e, it follows that $w(E') = w(M)$.

Since b-matching in bipartite graphs admits a polynomial time algorithm [18], it follows that Or can be solved in polynomial time. □

A similar reduction, implies that OrD has a 2-approximation algorithm.

Lemma 2. *The* OrD *problem has a* 2*-approximation algorithm.*

Proof. We use a reduction similar to that in Lemma 1. The only difference is that the edges in F inherit both the weight as well as demand of the corresponding edge, and in the bipartite graph H, we set $b_e = d_e$ for $e \in E$, and $b_v = c_v$ for each $v \in V$. Since demand-matching on bipartite graphs has a 2-approximation algorithm [16], the lemma follows. □

3 Related work

The PDBEP problem was introduced by Zhang [20] motivated by a problem of resource allocation as well as a problem of finding large independent sets. This is the unit demand and unit weight version of our problem, namely PEpD with the additional constraint that $w(e) = d_e = 1$ for all $e \in E$. As mentioned earlier, Zhang proved that unit capacity version of PDBEP i.e., PEp with unit edge weights is NP-Hard. This result follows from the fact that for a graph $G = (V, E)$, a solution of size k for PEp implies a dominating set of size $|V| - k$. Since the Dominating Set problem is NP-hard [12], this implies PEp is NP-hard. Zhang also presented a 2-approximation algorithm for PEp, with unit edge weights, and a 32/11-approximation algorithm, again with unit edge-weights and a uniform capacity of 2 for all vertices. Dehne et al. [8] studied a parameterized version of the PDBEP problem (with vertex capacity 1 and edge demands 1), and obtained algorithms that are exponential in the size of the PEp solution.

A related problem is the problem of packing vertex disjoint T-stars where a T-star is a complete bipartite graph $K_{1,t}$ for some $1 \leq t \leq T$. In [3], the authors gave a $\frac{9}{4}\frac{T}{T+1}$-approximation for this problem. When $T \geq |V| - 1$, the T-star packing in an edge weighted graph where the objective is to maximize the total weight of the edges in the stars is exactly the PEp problem.

Aurora et al. [2] presented a simple 2-approximation algorithm for PEp with arbitrary vertex capacity, but unit demands and unit weights on the edges. In the setting with weighted edges, but unit demands on the edges, they presented only an $O(\log n)$-approximation algorithm. We introduce the version of the PEp problem with demands on the edges, and present the first constant-factor approximation for this general case.

The PDBEP problem on hypergraphs was first studied by Aurora et al. [2], motivated by its natural application in the resource allocation. They presented a constant factor approximation algorithm for the r-PDBEP on k-uniform hypergraphs if both k and r are constants. The r-PDBEP problem is a generalization of the PDBEP problem, where each hyperedge in the solution demands the degree condition to be satisfied at at least r vertices it is incident on. We give a simple combinatorial d-approximation algorithm for the weighted PDBEP problem on hypergraphs if the maximum degree of any vertex is d.

The PEp problem, as stated earlier is similar to the Maximum weight matching problem, for which several polynomial time algorithms are known [10, 14]. The demand version of the problem, PEpD is similar to the *demand matching* problem introduced by Shepherd and Vetta [19]. For the demand matching problem, Shepherd and Vetta presented a 3.264-approximation for general graphs and a 2.764-approximation for bipartite graphs. These results have since been improved using the technique of iterative rounding to a factor 3 for general graphs, and a factor of 2 for bipartite graphs [16, 15].

The degree-bounded orienting problem Or is a classic combinatorial optimization problem. However, it has mostly received attention in terms of maintaining connectivity. See [18] for more details.

4 A 2-approximation algorithm for unit capacity instances

In this section, we present a 2-approximation algorithm for the PEp problem. In this setting, recall that $c_v = 1$ for all $v \in V$. Earlier, a 2-approximation algorithm was known only for the unweighted case by Zhang [20], i.e., when $w(e) = 1$ for all $e \in E$.

Our algorithm is combinatorial. We show that we can carefully select a subset of edges such that an upper bound on OPT_{PEp} can be constructed from this subset. Recall that for a vertex $v \in V$, $\delta_{\max}(v)$ denotes the set of edges of maximum weight incident on v, and we use $e_{\max}(v)$ to denote an edge in $\delta_{\max}(v)$.

The set of edges $E_{\max} \subseteq E$ is constructed as follows: Let $v_1, \ldots, v_n$ be an arbitrary ordering of the vertices. Starting with $E^1_{\max} = E^2_{\max} = \emptyset$, for each i from $1, \ldots, n$, if an edge from $\delta_{\max}(v_i)$ has not been chosen, pick an arbitrary edge $e = \{v_i, v_j\}$ from $\delta_{\max}(v_i)$. If $e \in \delta_{\max}(v_j)$ and no edges from $\delta_{\max}(v_j)$ have been chosen yet we add e to $E^2_{\max}$, otherwise add e to $E^1_{\max}$. We denote the set $E_{\max} = E^1_{\max} \cup E^2_{\max}$. Observe that by the way we choose the edges in $E^2_{\max}$

at most one edge from $\delta_{\max}(v)$ is chosen for each vertex. This is encoded in the Proposition below.

Proposition 2. *For each $v \in V$, $|E^2_{\max} \cap \delta_{\max}(v)| \leq 1$*

Lemma 3. $\mathrm{OPT}_{\textsc{PEp}} \leq \sum_{v \in V} w(e_{\max}(v))$

Proof. We show that $\sum_{v \in V} w(e_{\max}(v))$ is an upper-bound on the Max-orientation problem $\textsc{Or}$ on the same instance. Then, the lemma follows from Proposition 1. Let $F \subseteq E$ be a feasible solution to the $\textsc{Or}$ problem in G. Since the vertices have unit capacity, for any vertex $v \in V$, there is at most one edge of F in-coming to v. Let $w(in(v))$ denote the weight of this edge if any, and zero otherwise. Then, $\sum_{(u,v) \in F} w(u,v) = \sum_{v \in V} w(in(v)) \leq \sum_{v \in V} w(e_{\max}(v))$. □

Note that in order to obtain an upper bound, we require that we sum up $w(e_{\max}(v))$ over all vertices as $w(E_{\max})$ by itself does not constitute an upper bound. The example in Figure 1 shows this.

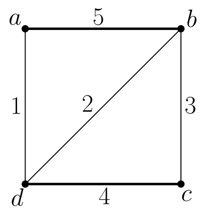

Fig. 1. The set $E_{\max}$ of edges is shown in bold. In this example, $\sum_{e \in E_{\max}} w(e) = w(a,b) + w(d,c) = 9$. However, $\mathrm{OPT}_{\textsc{PEp}} = w(a,b) + w(b,c) + w(b,d) = 10$.

In order to obtain our claimed approximation, we construct an orientation of the edges in $E_{\max}$ such that $\overrightarrow{G}_{\max} = (V, \overrightarrow{E}_{\max})$ is acyclic, and show that each connected component of $\overrightarrow{G}_{\max}$ is an out-tree.

Lemma 4. *There exists an orientation of the edges in $E_{\max}$ such that each connected component of $\overrightarrow{G}_{\max} = (V, \overrightarrow{E}_{\max})$ is an out-tree.*

Proof. We suggest a natural orientation of the edges in $E_{\max}$. Suppose the vertices are considered in the order of $v_1, \ldots, v_n$ during the construction of $E_{\max}$. In iteration i of the construction, if an edge is added to either $E^1_{\max}$ or $E^2_{\max}$, then orient that edge towards v_i. Since in each iteration at most one edge can be added to $E_{\max}$, our orientation ensures that for any vertex $v_i \in V$, at most one edge in $\overrightarrow{E}_{\max}$ is oriented towards v_i, which implies the in-degree of any vertex is at most 1. Note that this shows the set $E_{\max}$ is in fact a feasible solution to the $\textsc{Or}$ problem.

To show that each connected component of $\overrightarrow{G}_{\max}$ is an out-tree, we also require that the graph $\overrightarrow{G}_{\max}$ is acyclic. In contrary, let $C = v_0, v_1, \ldots, v_k, v_0$ be a directed cycle in $\overrightarrow{G}_{\max}$. Our orientation ensures that if an edge e is oriented towards v, then $e \in \delta_{\max}(v)$. Thus, the weight of an in-coming edge into a vertex v has weight at least as large as any out-going edge. Thus, following the cycle from v_0, the weight of the edges can not increase. Therefore, the only possibility is that all edges in C have equal weight. However, we add an edge e incident on a vertex v into $E_{\max}$ only if no edge in $\delta_{max}(v)$ is present in $E_{\max}$ when v is processed, and we then orient e towards v. If, without loss of generality, the vertex v_0 was the first to be processed during the construction of $E_{\max}$, then the edge $\{v_k, v_0\}$ is already present in the solution when we process v_k. Since $\{v_k, v_0\} \in \delta_{\max}(v_k)$, no edge is added during the processing of v_k. This implies no edge is oriented towards v_k, which is a contradiction. Hence, $\overrightarrow{G}_{max}$ is acyclic.

Thus, we have $\overrightarrow{G}_{\max}$ is acyclic and each vertex has in-degree at most 1. This ensures each connected component of $G_{\max}$ is an out-tree. □

The example in Figure 1 shows that $w(E_{\max})$ is not an upper bound on $\mathrm{OPT}_{\mathrm{PEP}}$. However, we can re-write the upper bound by distinguishing the contribution from the edges of $E^1_{\max}$ and $E^2_{\max}$.

Lemma 5. $\mathrm{OPT}_{\mathrm{PEP}} \leq w(E^1_{\max}) + 2w(E^2_{\max})$

Proof. Consider the above oriented graph $\overrightarrow{G}_{\max} = (V, \overrightarrow{E}_{\max})$. We claim that for any vertex v_i, if no edge is oriented towards v_i, then there is exactly one edge in $E^2_{\max}$ incident on v_i. To see this, note that if no edge is oriented towards v_i, then before iteration i, we have already added at least one edge from $\delta_{\max}(v_i)$ to $E_{\max}$. Let j be the minimal iteration such that an edge from the set $\delta_{\max}(v_i)$ is added to $E_{\max}$. Then according to our construction, the edge $\{v_j, v_i\}$ is the only edge added to the set $E^2_{\max}$ among all the edges incident on v_i in $E_{\max}$.

Let v be any vertex towards which no edge is oriented in $\overrightarrow{G}_{\max}$. Then there is exactly one edge in $E^2_{\max}$ incident on v. Let $e = \{u, v\}$ be that edge. We add another edge of weight $w(e)$ between u and v to $\overrightarrow{G}_{\max}$, and orient this edge towards v. Let the resulting multi-graph be $\overrightarrow{G}'_{\max} = (V, \overrightarrow{E}'_{\max})$. In $\overrightarrow{G}'_{\max}$, exactly one edge from $\delta_{\max}(v)$ is oriented towards v for each vertex v. Thus, $\sum_{v \in V} w(e_{\max}(v)) = \sum_{e \in \overrightarrow{E}'_{\max}} w(e) = \sum_{e \in E^1_{\max}} w(e) + 2\sum_{e \in E^2_{\max}} w(e)$. Using Lemma 3, we get $\mathrm{OPT}_{\mathrm{PEP}} \leq w(E^1_{\max}) + 2w(E^2_{\max})$. □

Theorem 1. *There exists a* 2*-approximation algorithm for* PEP.

Proof. Since each connected component of the graph $\overrightarrow{G}_{\max}$ is an out-tree, we can partition the vertices into two sets such that all the edges in $\overrightarrow{E}_{\max}$ cross the partition. To see this, for any tree T in $\overrightarrow{G}_{\max}$, label the vertices with distance from the root. Since T is a tree, a vertex with odd label is only adjacent to vertices with even label, and vice-versa. Therefore, we can partition V into two sets X and Y, where X consists of odd-labeled vertices, and Y consists of even-labeled vertices. The set of edges between X and Y consists of all the edges.

Now consider the cut (X, Y) in $G_{\max}$, and orient the edges in $E_{\max}$ as follows: Orient all the edges in $E^1_{\max}$ in the same way it is oriented in $\overrightarrow{G}_{\max}$, and orient each edge $e \in E^2_{\max}$, towards both the end-points. Note that $\sum_{e \in \delta^+(X)} w(e) + \sum_{e \in \delta^+(Y)} w(e) = w(E^1_{\max}) + 2w(E^2_{\max})$, since each edge in $E^2_{\max}$ is present in both $\delta^+(X)$ and $\delta^+(Y)$, where for any Z, $\delta^+(Z)$ denotes the set of out-going edges from Z. Therefore, $\max\{\delta^+(X), \delta^+(Y)\} \geq (w(E^1_{\max}) + 2w(E^2_{\max}))/2$. Thus, by returning the maximum among $\delta^+(X)$, and $\delta^+(Y)$, we guarantee a solution of weight at least $\mathrm{OPT}_{\mathrm{PEP}}/2$ (Using Lemma 5).

Now it remains to prove the feasibility of $\delta^+(X)$, and $\delta^+(Y)$. Note that the in-degree of any vertex is at most 1 in the oriented graph, which ensures both $\delta^+(X)$ and $\delta^+(Y)$ are individually feasible for PEP. □

Note that our 2-approximation algorithm for PEP improves the result of Babenko and Gusakov [3] for the special case of T-star packing when $T = |V| - 1$. They proposed a $\frac{9}{4}\frac{T}{T+1}$-approximation algorithm for this problem.

5 A constant-factor approximation algorithm

In this section, we obtain a $(4+\epsilon)$-approximation algorithm for any $\epsilon > 0$ for the PEPD problem on general graphs, and a $(2+\epsilon)$-approximation algorithm for the PEPD problem on bipartite graphs. Our algorithm holds for a slightly more general problem. Instead of demands on the edges, we let each edge $e = \{u, v\}$ have possibly different demands $d(e,u)$, $d(e,v)$ at its end-points. It is possible that $d(e,u)$ exceeds the capacity c_u of vertex u, and yet, e could be in our solution since its other end-point, namely v could be its good end-point.

Given a graph $G = (V, E)$, our algorithm finds an oriented multi-graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ having $w(E')$ almost equal to the optimal PEPD solution such that for all the vertices, the in-degree constraint is satisfied. Next, by finding a directed cut of weight at least $w(E')/4$ in $\overrightarrow{G}'$, we guarantee a $(4+\epsilon)$-approximate solution for PEPD in G.

Lemma 6. *Given a graph $G = (V, E)$, weights $w : E \to \mathbb{N}$, demands $d : E \to \mathbb{N}$ on the edges, and capacities $c : V \to \mathbb{N}$ on the vertices, a directed multi-graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ with $w(E')$ at least $(1-\epsilon)\mathrm{OPT}_{\mathrm{PEPD}}$ can be constructed from G in polynomial time such that $\sum_{e \in in(v)} d_e \leq c_v$, for all $v \in V$.*

Proof. Let $\mathrm{OPT}_{\mathrm{PEPD}}$ denote an optimal solution for the PEPD problem, and $F \subseteq E$ be the edges picked in this solution. For any vertex v_i, if the degree condition is satisfied in F, then we set $\mathrm{OPT}^i_{\mathrm{PEPD}}$ to be the total weight of the edges incident on v_i in F, otherwise we set $\mathrm{OPT}^i_{\mathrm{PEPD}}$ to be 0. Since for any edge in F, the degree condition is satisfied at at least one end-point, we have $\sum_{i=1}^{n} \mathrm{OPT}^i_{\mathrm{PEPD}} \geq \mathrm{OPT}_{\mathrm{PEPD}}$.

For all $v_i \in V$, we consider the problem of picking a maximum weight sub-set of edges from $\delta(v_i)$ such that the degree condition is satisfied. Observe that at each vertex, this amounts to solving an independent Knapsack problem. Since Knapsack admits an FPTAS [17], we obtain a solution $\mathcal{A}_i$ of weight at least

$(1-\epsilon)\mathrm{OPT}_i$, where OPT_i is the optimal solution to this problem w.r.t. vertex v_i. Therefore, we have

$$\begin{aligned}\sum_{i=1}^{n} w(\mathcal{A}_i) &\geq (1-\epsilon)\sum_{i=1}^{n}\mathrm{OPT}_i \\ &\geq (1-\epsilon)\sum_{i=1}^{n}\mathrm{OPT}^i_{\textsc{PEpD}} \\ &\geq (1-\epsilon)\mathrm{OPT}_{\textsc{PEpD}}.\end{aligned}$$

The second inequality is true because $\mathrm{OPT}^i_{\textsc{PEpD}}$ is a feasible solution to the knapsack problem w.r.t. v_i.

Observe that in a similar way, we can show that $\sum_{i=1}^{n} w(\mathcal{A}_i)$ is at least $(1-\epsilon)\mathrm{OPT}_{\textsc{OrD}}$. The set of edges in $\cup_{i=1}^{n}\mathcal{A}_i$ is in fact a feasible solution to the OrD problem. To see this, for each vertex v_i, orient the edges in $\mathcal{A}_i \setminus \cup_{j=1}^{i-1}\mathcal{A}_j$ towards the vertex v_i. This ensures that for any vertex, the total demand of the incoming edges is at most its capacity. Since, each edge can appear at most twice in the sum $\sum_{i=1}^{n} w(\mathcal{A}_i)$, we have a $(2+\epsilon)$-approximation algorithm for the OrD problem.

However, the set of edges in $\cup_{i=1}^{n}\mathcal{A}_i$ may not be a feasible solution for the PEpD problem. In order to obtain the $(4+\epsilon)$-approximation, we construct a directed multi-graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ as follows: Pick an arbitrary ordering of the vertices, say $v_1, \ldots, v_n$. Starting with $E' = \emptyset$, for each i from $1, \ldots n$, add the edges in $\mathcal{A}_i$ to the multi-set of edges E' and orient the edges in $\mathcal{A}_i$ towards v_i. By doing this, we ensure that $\sum_{e\in E'} w(e) = \sum_{i=1}^{n} w(\mathcal{A}_i) \geq (1-\epsilon)\mathrm{OPT}_{\textsc{PEpD}}$. □

Theorem 2. *For any $\epsilon > 0$, there exists a $(2+\epsilon)$-approximation algorithm for* PEpD *on bipartite graphs.*

Proof. Given a bipartite graph $G = (U \cup V, E)$, using Lemma 6, we can find the directed multi-graph $\overrightarrow{G}' = (U \cup V, \overrightarrow{E}')$ with $w(E') \geq (1-\epsilon')\mathrm{OPT}_{\textsc{PEpD}}$, for any $\epsilon' > 0$ such that total demand of the in-coming edges to any vertex is at most its capacity. So, the set of incoming edges to U and the set of in-coming edges to V in $\overrightarrow{G}'$ are separately feasible for the PEpD problem on G, and the maximum of both has weight at least $(1-\epsilon')\mathrm{OPT}_{\textsc{PEpD}}/2$. Choosing $\epsilon' = \epsilon/(2+\epsilon)$, we obtain a solution of weight at least $\mathrm{OPT}_{\textsc{PEpD}}/(2+\epsilon)$ for the PEpD problem on bipartite graphs. □

In order to get the desired approximation ratio for general graphs, we find a directed cut (DICUT) of weight at least $w(E')/4$ in $\overrightarrow{G}'$. Given a directed multi-graph $\overrightarrow{G}_m$ and an edge weight function $w : E(G_m) \to \mathbb{N}$, a DICUT is defined to be the set of out-going edges from some vertex subset X (we denote it by $\delta^+(X)$). Note that any directed cut in $\overrightarrow{G}'$ is a feasible PEpD solution. Lemma 7 captures this.

Lemma 7. *Any directed cut of the directed multi-graph $\overrightarrow{G}'$ is a feasible* PEpD *solution of G.*

Proof. Let $\delta^+(X)$ be a DICUT of $\overrightarrow{G}'$. This implies for all $v \in V \setminus X$, $out(v) = \emptyset$, and for any edge (u, v) in $\delta^+(X)$ directed towards v, the degree condition of v is satisfied in G. This ensures $\delta^+(X)$ is a feasible PEpD solution. □

Lemma 8. *Given a directed multi-graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$, weights $w : E' \to \mathbb{N}$, there exists a directed cut of size at least $w(E')/4$.*

Proof. Consider the trivial randomized algorithm which adds any vertex in V to the set X with probability 1/2. Any directed edge $e = (u, v)$ is a cut if $u \in X$ and $v \in V \setminus X$. This happens with probability 1/4. So, the expected weight of the DICUT is

$$\mathbb{E}\Big(\sum_{e\in\delta^+(X)} w(e)\Big) = \sum_{e\in\overrightarrow{E}'} w(e) \cdot \Pr\big(e \in \delta^+(X)\big) = \sum_{e\in\overrightarrow{E}'} w(e) \cdot \frac{1}{4} = \frac{w(E')}{4}.$$

This ensures, there exists a DICUT of weight at least $w(E')/4$. To find it, derandomize this by using the method of conditional expectations. □

Armed with Lemma 6, 7, and 8 we can now complete the proof.

Theorem 3. *There exists a $(4+\epsilon)$-approximation algorithm for* PEpD*, for any $\epsilon > 0$.*

Proof. Given an instance of PEpD, let OPT_{PEpD} be an optimal solution to PEpD. Lemma 6 shows that we can obtain an oriented graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ having $w(E') \geq (1 - \epsilon')\text{OPT}_{\text{PEpD}}$, for any $\epsilon' > 0$ such that $\sum_{e\in in(v)} d_e \leq c_v$, for all $v \in V$. Combining this with Lemma 7 and Lemma 8 we obtain a PEpD solution in G of weight at least $w(E')/4$ which is at least $(1 - \epsilon')\text{OPT}_{\text{PEpD}}/4$. Using $\epsilon' = \epsilon/(4+\epsilon)$, we get a PEpD solution of weight at least $\text{OPT}_{\text{PEpD}}/(4+\epsilon)$. □

6 A constant approximation for a generalized PEpD problem

In this section, we obtain a $(7 + \epsilon)$-approximation algorithm for a generalized PEpD problem in which we allow each edge to decide when it is satisfied. In this problem, some edges only require that the degree condition is satisfied at at least one end-vertex while the remaining edges require that the degree condition at both end-vertices be satisfied. Formally, given an undirected graph $G = (V, E_1 \cup E_2)$ with $w, d : (E_1 \cup E_2) \to \mathbb{N}$, the weights and demands respectively on the edges, and $c : V \to \mathbb{N}$ the capacity on the vertices. The goal is to find a subgraph $G' = (V, E')$ such that $w(E')$ is maximized, and each edge in E' is satisfied, where any edge $e = (u, v) \in E_1$ is satisfied if $d(\delta'(u)) \leq c_u$ or $d(\delta'(v)) \leq c_v$, and any edge $e = (u, v) \in E_2$ is satisfied if $d(\delta'(u)) \leq c_u$ and $d(\delta'(v)) \leq c_v$.

Note that this generalized PEpD problem on the subgraph $G_1 = (V, E_1)$ is the same as the PEpD problem on G_1 and this problem on the subgraph $G_2 =$

(V, E_2) is the same as the demand matching problem on G_2. Our algorithm solves the PEpD problem on G_1 and the demand-matching problem on G_2 and then returns the solution of maximum weight. Since we can have a $(4+\epsilon)$-approximate solution for the PEpD problem on G_1 and a 3-approximate solution for the demand matching problem on G_2, we have a $(7+\epsilon)$-approximate solution for this generalized PEpD on G. To see this, let OPT be the optimal solution to the generalized PEpD problem on G and OPT_1, OPT_2 be the optimal solutions to the PEpD problem on G_1 and the demand matching problem on G_2 respectively. Let $A_1 \geq \mathrm{OPT}_1/(4+\epsilon)$ be any solution to PEpD on G_1 and $A_2 \geq \mathrm{OPT}_2/3$ be any solution to the demand matching problem on G_2. We have

$$\begin{aligned} \mathrm{OPT} &\leq \mathrm{OPT}_1 + \mathrm{OPT}_2 \\ &\leq (4+\epsilon)A_1 + 3A_2 \\ \implies \max\{A_1, A_2\} &\geq \frac{\mathrm{OPT}}{7+\epsilon}. \end{aligned}$$

Theorem 4. *For any $\epsilon > 0$, there exists a $(7+\epsilon)$-approximation algorithm for the generalized* PEpD *problem in which some edges are satisfied if the degree condition is satisfied at at least one end-vertex while the other edges are satisfied if degree condition is satisfied at both end-vertices.*

7 A constant approximation for PDBEP on hypergraphs of constant degree

In this section, we study the PDBEP problem on hypergraphs. We present a d-approximation algorithm for the weighted PDBEP problem on hypergraphs having maximum degree d. Given a hypergraph $H = (X, E)$ with maximum degree d. We partition E into at most d disjoint subsets $\{E_1, E_2, \cdots, E_d\}$ such that each subset E_i is a feasible PDBEP solution. So, the subset of edges with maximum weight gives a d-approximate solution. We construct this partition using Algorithm 1.

Algorithm 1 Edge Partitioning

```
$i \leftarrow 1$
while $E \neq \emptyset$ do
  $E_i \leftarrow E$
  while $\exists$ unsatisfied edge $e \in E_i$ do
    $E_i \leftarrow E_i \setminus e$
  end while
  $E \leftarrow E \setminus E_i$
  $i \leftarrow i + 1$
end while
```

Lemma 9. *Algorithm* 1 *partitions E into at most d subsets and each subset E_i is a feasible solution to the PDBEP problem.*

Proof. It is easy to verify that each E_i is a feasible solution to the PDBEP problem, otherwise there exists an unsatisfied edge in E_i.

At any iteration i, we remove the unsatisfied hyperedges from E_i and if any hyperedge e is not satisfied then all the vertices on which e is incident have degree at least 2. Hence, in the sub-hypergraph $H_i = (X_i, E_i)$, each vertex has degree at least 1, where $X_i \subseteq X$ is the set of vertices whose degree is at least 1 in the residual hypergraph before iteration i starts. This implies if a vertex has degree at least 1 in the residual hypergraph at the end of iteration $(i-1)$, then its degree decreased by at least 1 at the end of iteration i. Since the degree of any vertex is at most d, after d rounds every vertex has degree 0. So, algorithm 1 partitions E into at most d subsets. $\square$

Theorem 5. *The weighted PDBEP problem on hypergraphs having maximum degree d has a d-approximation algorithm.*

Proof. Let OPT be the optimal solution to the PDBEP problem on $H = (X, E)$ and Algorithm 1 partition E into d subsets $E_1, E_2, \cdots, E_d$. We have,

$$\sum_{i=1}^{d} w(E_i) = w(E) \geq w(\mathrm{OPT})$$

$$\implies \max_i w(E_i) \geq \frac{w(\mathrm{OPT})}{d}$$

Thus by returning the maximum weighted subset among $\{E_1, E_2, \cdots, E_d\}$, we get a d-approximate solution. $\square$

8 PTAS for PEpD on minor-free graphs

In this section, we obtain a PTAS for PEpD on H-minor-free graphs. Our result follows the standard procedure for proving a PTAS for such graphs. We present a polynomial time algorithm for graphs of bounded-treewidth. However, the algorithm only works in the setting where the demands on the edges are bounded by a polynomial. A PTAS for H-minor-free graphs then follows from the results of Demaine et al. [9].

8.1 A polynomial time algorithm for bounded-treewidth graphs

For the sake of completeness, we give a definition of a tree-decomposition. See [5] for a description and results on tree-decompositions.

Definition 1. *A tree decomposition of a graph $G = (V, E)$ is a pair (T, X), where $T = (I, F)$ is a tree and $X = \{X_i |\ i \in I\}$ is a set with $X_i \subseteq V$ satisfying*

– $\bigcup_{i \in I} X_i = V$.
– *for any edge $e = (u, v) \in E$, there exists an $i \in I$ with $u \in X_i$ and $v \in X_i$.*
– *for all $v \in V$, the set of nodes $\{i \in I |\ v \in X_i\}$ forms a connected subtree of T.*

We refer to the vertices of T as nodes and the corresponding X_i's as bags in order to distinguish them from the vertices of G. The width of any tree decomposition $T = (I, F)$ is $\max_{i \in I} |X_i| - 1$ and the tree-width of a graph G, denoted as $tw(G)$ is the minimum width among all possible tree decompositions of G. Let G be a graph with $tw(G) = t - 1$, for constant $t > 0$, and let (T, X) with $T = (I, F)$ and $X = \{X_i |\ i \in I\}$ be a tree decomposition of G of width $t - 1$. It is also well known that without loss of generality we can assume that T is a rooted binary tree [5].

Define for all $i \in I$, $Y_i = \{v \in X_j |\ j \text{ is a proper descendant of } i\}$. Let $G[X_i] = (X_i, E_i)$, and $G[Y_i] = (Y_i, F_i)$ denote the vertex induced subgraphs of G with vertices in X_i and Y_i respectively. Let $G' = (V, E')$ be an optimal solution for PEpD, and $F'_i = F_i \cap E'$, $E'_i = E_i \cap E'$. For any $v \in X_i$, let $\text{dem}_i(v)$ be the total demand of the edges incident on v in the subgraph $G'_i = (V, F'_i \cup E'_i)$. For any bag X_i, let $\mathbf{d_i}$ be the vector representing $\text{dem}_i(v)$ for each vertex $v \in X_i$. Suppose $\mathbf{f}$ be another vector representing whether $\text{dem}(v) \le c_v$, or $\text{dem}(v) > c_v$, where $\text{dem}(v)$ denotes the total demand of the edges incident on v in G'. For any $v \in V$, we set $\mathbf{f}(v) = c_v$, if $\text{dem}(v) \le c_v$, and $\mathbf{f}(v) = \infty$, otherwise. The vector $\mathbf{f_i}$ denotes the vector $\mathbf{f}$ with restriction to the vertices in X_i.

We now describe our dynamic program. The dynamic program works bottom-up. Each DP cell $\mathsf{C}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})$ represents the subproblem of choosing a set of edges $F'_i \subseteq F_i$ with maximum total weight, such that $F'_i \cup E'_i$ are feasible assuming $\text{dem}_i(v)$ and $\text{dem}(v)$ are bounded above by $\mathbf{d_i}$ and $\mathbf{f_i}$ respectively.

For any vertex $v \in X_i$, $\mathbf{f}(v)$ is either c_v or ∞. Since $\mathbf{d_i} \preceq \mathbf{f_i}$ i.e., the vector $\mathbf{d_i}$ is component-wise less than or equal to the vector $\mathbf{f_i}$, $\mathbf{d_i}(v)$ can take values in $\{0, 1, \cdots, c_v\}$ if $\mathbf{f}(v) = c_v$ and can take values in $\{0, 1, \cdots, d(\delta(v))\}$ if $\mathbf{f}(v) = \infty$, where $d(\delta(v))$ denotes the total demand of the edges in E incident on v. For each node i, we enumerate over all subsets of E_i to find E'_i, and enumerate over all possible vectors that $\mathbf{f_i}$ and $\mathbf{d_i}$ can take to find $\mathbf{f_i}$ and $\mathbf{d_i}$. Since $\mathbf{f}$ provides an upper bound on the demand on any vertex in G', edges in any subgraph of G' must satisfy the feasibility constraint assuming demand on any vertex v can be at most $\mathbf{f}(v)$. Note that for any node $v \in X_i$, $\mathbf{d_i}(v)$ is the demand of the edges in $F'_i \cup E'_i$ incident on v. So, for any leaf node i, we set $\mathbf{d_i}(v) = \text{dem}_{E'_i}(v)$, where $\text{dem}_{E'_i}(v)$ represents the demand of the edges in E'_i incident on v. Let $\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})$ be the weight of the DP cell $\mathsf{C}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})$. For any leaf node i, we compute $\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})$ as follows:

$$\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i}) = \begin{cases} w(E'_i), & \text{if } \forall \{u, v\} \in E'_i,\ \mathbf{f_i}(u) \le c_u \text{ or } \mathbf{f_i}(v) \le c_v \\ -\infty, & \text{otherwise.} \end{cases}$$

For any internal node i, with children j and k for which we have already computed the DP cells, we can compute the DP cell as follows:

$$\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i}) = \begin{cases} \mathcal{A}\big(\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})\big), & \text{if } \forall\{u,v\} \in E'_i,\ \mathbf{f_i}(u) \le c_u \text{ or } \mathbf{f_i}(v) \le c_v \\ -\infty, & \text{otherwise.} \end{cases}$$

Where, $\mathcal{A}\big(\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})\big)$ can be computed as follows:

$$\begin{aligned} \mathcal{A}\big(\mathsf{W}(i, E'_i, \mathbf{f_i}, \mathbf{d_i})\big) = \max_{\substack{E'_j, \mathbf{f_j}, \mathbf{d_j} \preceq \mathbf{f_j}, \\ E'_k, \mathbf{f_k}, \mathbf{d_k} \preceq \mathbf{f_k}}} \Big\{ & \mathsf{W}(j, E'_j, \mathbf{f_j}, \mathbf{d_j}) + \mathsf{W}(k, E'_k, \mathbf{f_k}, \mathbf{d_k}) \\ & + w(E'_i \setminus (E'_j \cup E'_k)) - w(E'_j \cap E'_k) \Big| \\ & \forall v \in \{X_i \cap X_j \cap X_k\}, \mathbf{f_i}(v) = \mathbf{f_j}(v) = \mathbf{f_k}(v), \\ & \forall v \in \{X_i \cap X_j\}, \mathbf{f_i}(v) = \mathbf{f_j}(v), \\ & \forall v \in \{X_i \cap X_k\}, \mathbf{f_i}(v) = \mathbf{f_k}(v), \\ & \forall v \in X_i, \mathbf{d_i}(v) \ge \mathbf{d_j}(v) + \mathbf{d_k}(v) + \\ & \mathrm{dem}_{E'_i \setminus (E'_j \cup E'_k)}(v) - \mathrm{dem}_{E'_j \cap E'_k}(v) \Big\}, \end{aligned}$$

Where, $\mathrm{dem}_{E'_i \setminus (E'_j \cup E'_k)}(v)$ and $\mathrm{dem}_{E'_j \cap E'_k}(v)$ denote the demands of the edges incident on v in the subgraphs $(V, E'_i \setminus (E'_j \cup E'_k))$ and $(V, E'_j \cap E'_k)$ respectively. Note that if $v \notin X_j$, then $\mathbf{d_j}(v) = 0$ and if $v \notin X_k$, then $\mathbf{d_k}(v) = 0$.

The optimal solution is the $\max_{E'_r, f_r}\{\mathsf{W}(r, E'_r, \mathbf{f_r}, \mathbf{f_r})\}$, where r is the root node of the tree T.

The number of nodes in the tree-decomposition T of G is at most $O(nt)$ [5]. For any node $i \in I$, $|X_i| \le t$, so E'_i can take at most 2^{t^2} values, and f_i can take at most 2^t values. For any vertex $v \in X_i$, $\mathbf{d_i}(v)$ can take at most $nd_{\max}$ values, where $d_{\max} = \max_e d_e$. So, total number of cells in the DP can be at most $O(nt) \cdot 2^{t^2} \cdot 2^t \cdot (nd_{\max})^t = O(n(nd_{\max})^t)$. Each DP cell takes $O((nd_{\max})^{2t})$ computation time. So the running time of the DP is $O(n(nd_{\max})^{3t})$ which is polynomial if $d_{\max}$ is bounded above by a ploynomial.

Theorem 6. *The* PEPD *problem, on graphs with bounded tree-width can be solved in polynomial time if the demands on the edges are bounded by a polynomial.*

8.2 Partition into bounded treewidth graphs

We use the following result of Demaine et al. [9] on the structure of H-minor-free graphs.

Lemma 10 ([9]). *For a fixed graph H, there is a constant c_H such that, for any integer $k \ge 1$ and for every H-minor-free graph G, the vertices of G (or the* *edges of G) can be partitioned into $k+1$ sets such that any k of the sets induce a graph of tree-width at most $c_H k$. Furthermore, such a partition can be found in polynomial time .*

Theorem 7. *In H-minor free graphs, there is a PTAS for the* PEpD *problem if the demands on the edges are bounded by polynomial.*

Proof. Let $G = (V, E)$ be any H-minor-free graph. We apply Lemma 10 with $k = 1/\epsilon$ to partition E into sets $E_1, E_2, \cdots E_{1+1/\epsilon}$. Let E' be the edges in the optimal solution, and $E'_1 = E' \cap E_1, E'_2 = E' \cap E_2, \cdots, E'_{1+1/\epsilon} = E' \cap E_{1+1/\epsilon}$. Let E'_m be the set with minimum weight among $\{E'_1, E'_2, \cdots, E'_{1+1/\epsilon}\}$. Since, $w(E'_m) \leq \frac{w(E')}{k+1}$. this implies $w(E' \setminus E'_m) \geq (1 - 1/(k+1))w(E')$.

Let G_i be the subgraph of G with edge set $\overline{E}_i = \cup_{j \neq i} E_j$. Each G_i has tree-width bounded by $c_H k$ for which we can get the optimal solution OPT_i by using Theorem 6. Since the set $E' \setminus E'_i$ is a feasible solution to G_i, we have $\mathrm{OPT}_i \geq w(E' \setminus E'_i)$.

$$\begin{aligned} \max\{\mathrm{OPT}_1, \cdots, \mathrm{OPT}_{k+1}\} &\geq \mathrm{OPT}_m \geq w(E' \setminus E'_m) \\ &\geq \left(1 - \frac{1}{k+1}\right) w(E') \geq (1-\epsilon) w(E'). \end{aligned}$$

Hence, the maximum weighted solution among the solutions for $G_1, G_2, \cdots G_{k+1}$ gives a PTAS. □

9 APX-hardness

In this section, we prove that the PEp problem is APX-hard even for unweighted bipartite graphs of degree bounded by 3. Earlier, only NP-hardness was known. This was proved by Zhang [20] by showing that for a graph $G = (V, E)$, a solution to unweighted PEp of size k implies a Dominating Set in G of size $|V| - k$. Our result follows from the following facts: graphs of bounded degree have large dominating sets, and the fact that the Dominating Set problem is APX-hard on bipartite graphs of degree bounded by 3.

Proposition 3. *Let $G = (V, E)$ be a graph with degree bounded by B. Then, any dominating set in G has size at least $|V|/(1 + B)$.*

Proof. Let $|V| = n$ and suppose there is a dominating set $S \subset V$ of size $< n/(1 + B)$. Since each $v \in S$ can dominate at most B vertices in $V \setminus S$, all vertices in S together can dominate $< nB/(B + 1)$ vertices. Since any vertex in G either belongs to S or is dominated by a vertex in S, we have the total number of vertices in $G < n/(B + 1) + nB/(B + 1) = n$ contradicting our assumption that $|V| = n$. □

Lemma 11. *The Dominating Set problem on bipartite graphs of degree bounded by 3 is APX-hard.*

Proof. It is known that the dominating set problem on cubic graphs is APX-hard [1]. We reduce this problem to the dominating set problem on bipartite graphs of degree bounded by 3. We use the reduction identical to the one given in [11], which we describe here for completeness. Let $G = (V, E)$ be a cubic graph. We reduce G to a bipartite graph $G' = (X \cup Y, E')$ with maximum degree 3 as follows: For each edge $e = \{x, y\} \in E(G)$, we subdivide e by adding 3 new vertices $S_e = \{a_e, b_e, c_e\}$. Hence, $|E(G')| = 4|E(G)|$, and $E(G')$ contains 4 edges $\{x, a_e\}, \{a_e, b_e\}, \{b_e, c_e\}$ and $\{c_e, y\}$ corresponding to each edge $e \in E(G)$. Set $X = V \cup \{b_e : e \in E(G)\}, Y = \{a_e, c_e : e \in E(G)\}$. Clearly, G' is bipartite. The maximum degree in G' is 3, since the degree of each vertex in $V(G)$ remains unchanged, and the new vertices added have degree 2.

Let D' be a dominating set of G', i.e., for each $v \in V(G')$, either v or a neighbour u of v is in D', and we say that u *dominates* v. We claim that any dominating set D' of G' can be modified without increasing its size such that $|S_e \cap D'| = 1, \ \forall e \in E(G)$. For an edge $e \in E(G)$, either the vertex $b_e \in D'$, or it is dominated by at least one of its neighbours. Hence, $|S_e \cap D'| \geq 1$. Suppose $|D' \cap S_e| = 3$, then clearly $D' \setminus \{b_e\}$ is a dominating set of smaller size. Hence, we can assume $|D' \cap S_e| \leq 2$ for all $e \in E(G)$. Suppose there is an edge $e = \{x, y\} \in E(G)$ such that $|S_e \cap D'| = 2$. First, assume that $\{a_e, c_e\} \in D'$. If $x, y \in D'$, we can remove one of a_e or c_e to obtain a dominating set of smaller size. Therefore, we can assume that at least one of x, y is not in D'. Without loss of generality, assume $y \notin D'$. Then, $D'' = (D' \setminus \{c_e\}) \cup \{y\}$ is a dominating set, and $|D''| = |D'|$. If $\{a_e, b_e\} \in D'$, and if $y \in D'$, $D' \setminus \{b_e\}$ is a dominating set of smaller size. If $y \notin D'$, then $D'' = (D' \setminus \{b_e\}) \cup \{y\}$ is a dominating set, and $|D''| = |D'|$. Hence, for any dominating set D' of G', there is a dominating set D'' of G' such that $|D''| \leq |D'|$, and $|D'' \cap S_e| = 1, \ \forall e \in E(G)$.

Let D' be a dominating set of G' such that $|D' \cap S_e| = 1, \ \forall e \in E(G)$. Then, we claim that $D = V(G) \cap D'$ is a dominating set for G. To see this, consider a vertex $x \in V(G)$. Suppose $x \notin D$. Then, there is an edge $e = \{x, y\} \in E(G)$ such that $a_e \in D'$, since D' is a dominating set for G'. Since $|D' \cap S_e| = 1$, this implies $y \in D'$. Hence, x is dominated by $y \in D$. Therefore,

$$|D| \leq |D'| - |E| \tag{1}$$

On the other hand, let D be a dominating set in G. Then, we construct a dominating set D' for G' as follows: Suppose $u \in V(G)$ is not in D. Then, there is an edge $e = \{u, v\} \in E(G)$ such that $v \in D$. Add the vertex a_e in D'. Then, a_e dominates b_e and v dominates c_e. If $e = \{u, v\}$ is such that both $u, v \notin D$, or both $u, v \in D$. Then add the vertex b_e to D'. This ensures all vertices in G' are dominated and we have added one vertex for each edge in $E(G)$. Therefore,

$$|D'| = |D| + |E| \tag{2}$$

We now show that our reduction is an L-reduction. See [7] for definitions. Let OPT and OPT$'$ denote the optimal solutions for G and G' respectively. From Eqn.(2), it follows that $|\text{OPT}'| \leq |\text{OPT}| + |E|$. From Proposition 3, we have

$|\mathrm{OPT}| \geq |V|/4$, and since G is a cubic graph, we have $|E| \leq 3|V|/2$. Putting these together, we get

$$\begin{aligned}
|\mathrm{OPT}'| &\leq |\mathrm{OPT}| + |E| && \text{[eqn. (2)]} \\
&\leq |\mathrm{OPT}| + 3|V|/2 && \text{[G is a cubic graph]} \\
&\leq |\mathrm{OPT}| + 6|\mathrm{OPT}| && \text{[from Proposition (3)]} \\
&= 7|\mathrm{OPT}|
\end{aligned}$$

For any dominating set D' for G', let D be the corresponding dominating set for G obtained as above. From Eqn.(1), $|D| \leq |D'| - |E|$. Therefore,

$$\begin{aligned}
|D| - |\mathrm{OPT}| &\leq (|D'| - |E|) - (|\mathrm{OPT}'| - |E|) \\
&= |D'| - |\mathrm{OPT}'|
\end{aligned}$$

Hence, we have shown that the dominating set problem on bipartite graphs of degree bounded by 3 is APX-hard if the dominating set problem on cubic graphs is APX-hard. □

Next, we give a PTAS reduction from the dominating set problem on unweighted bipartite graphs of degree bounded by 3 to the PEP problem on unweighted bipartite graphs of degree bounded by 3.

We use the following result of Zhang [20] on the relation between Dominating Sets and PEP on unweighted graphs.

Lemma 12 ([20]). *Let $G = (V, E)$ be a graph without isolated vertices having $w(e) = 1$, for all $e \in E$. In G, there is a solution of size k to the* PEP *problem if and only if there is a solution of size $|V| - k$ to the dominating set problem.*

Theorem 8. *The* PEP *problem is APX-hard for unweighted bipartite graphs having degree at most* 3.

Proof. We prove that an existence of a PTAS for the PEP problem on bipartite graphs of degree at most 3 implies a PTAS for the dominating set problem on the same class of graphs, contradicting Lemma 11.

Let $G = (U \cup V, E)$ be a bipartite graph with degree bounded by 3 and $|U \cup V| = n$. By Proposition 3, $\mathrm{OPT}_{DS}(G) \geq n/4$, where $\mathrm{OPT}_{DS}(G)$ denotes the optimal solution to the dominating set problem on G. Lemma 12 implies that $\mathrm{OPT}_{\mathrm{PEP}}(G) = n - \mathrm{OPT}_{DS}(G)$, where $\mathrm{OPT}_{\mathrm{PEP}}(G)$ denotes the optimal solution to the PEP problem on G. Suppose there exists a PTAS for the PEP problem. This implies that for every $\epsilon > 0$, we can find a sub-graph $G' = (V, E')$ such that $|E'| \geq (1 - \epsilon)(n - \mathrm{OPT}_{DS}(G))$. By Lemma 12, there exists a dominating set C of size, $|C| = n - |E'|$. Therefore,

$$\begin{aligned}
|C| = n - |E'| &\leq n - (1 - \epsilon)(n - \mathrm{OPT}_{DS}(G)) \\
&\leq 4\epsilon \mathrm{OPT}_{DS}(G) + (1 - \epsilon)\mathrm{OPT}_{DS}(G) \\
&\leq (1 + 3\epsilon)\mathrm{OPT}_{DS}(G).
\end{aligned}$$

Therefore, the PEP problem is APX-hard, even on unweighted bipartite graphs of degree bounded by 3. □

10 Conclusion

To obtain better than 2-approximation for PEp, $(2+\epsilon)$-approximation for PEpD on bipartite graphs, and $(4+\epsilon)$-approximation for PEpD on general graphs, we need to find better upper bounds on OPT_{PEp}, OPT_{PEpD} on bipartite graphs, and OPT_{PEpD} on general graphs respectively.

Following example in Figure 2 shows that the upper bound is factor 2 away from the optimal PEp solution.

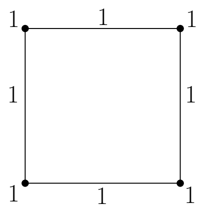

Fig. 2. $\text{OPT}_{\text{PEp}} = 2$, where the upper bound is $\sum_{v \in V} w(e_{\max}(v)) = 4$

References

1. Paola Alimonti and Viggo Kann. Hardness of approximating problems on cubic graphs. In *Algorithms and Complexity, Third Italian Conference, CIAC '97, Rome, Italy, March 12-14, 1997, Proceedings*, pages 288–298, 1997.
2. Pawan Aurora, Sumit Singh, and Shashank K. Mehta. Partial degree bounded edge packing problem for graphs and k-uniform hypergraphs. *J. Comb. Optim.*, 32(1):159–173, 2016.
3. Maxim A. Babenko and Alexey Gusakov. New exact and approximation algorithms for the star packing problem in undirected graphs. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 519–530, 2011.
4. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
5. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.
6. Tian-Ming Bu, Chen Yuan, and Peng Zhang. Computing on binary strings. *Theoretical Computer Science*, 562:122–128, 2015.
7. Pierluigi Crescenzi. A short guide to approximation preserving reductions. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity, Ulm, Germany, June 24-27, 1997*, pages 262–273, 1997.
8. Frank K. H. A. Dehne, Michael R. Fellows, Henning Fernau, Elena Prieto, and Frances A. Rosamond. NONBLOCKER: parameterized algorithmics for minimum dominating set. In *SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Merín, Czech Republic, January 21-27, 2006, Proceedings*, pages 237–245, 2006.

9. Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 637–646, 2005.
10. Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.
11. Jukka Suomela (https://cstheory.stackexchange.com/users/74/jukka suomela). Is the dominating set problem restricted to planar bipartite graphs of maximum degree 3 NP-complete? Theoretical Computer Science Stack Exchange. URL:https://cstheory.stackexchange.com/q/2508 (version: 2010-11-01).
12. Richard M Karp. *Reducibility among combinatorial problems.* Springer, 1972.
13. László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.
14. Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.
15. Ojas Parekh. Iterative packing for demand and hypergraph matching. In *Integer Programming and Combinatoral Optimization*, pages 349–361. Springer, 2011.
16. Ojas Parekh and David Pritchard. Generalized hypergraph matching via iterated packing and local ratio. In *Approximation and Online Algorithms*, pages 207–223. Springer, 2014.
17. Sartaj Kumar Sahni. On the knapsack and other computationally related problems. *Ph.D. Dissertation, Cornell University*, 1973.
18. Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
19. F. Bruce Shepherd and Adrian Vetta. The demand matching problem. In *Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings*, pages 457–474, 2002.
20. Peng Zhang. Partial degree bounded edge packing problem. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management - Joint International Conference, FAW-AAIM 2012, Beijing, China, May 14-16, 2012. Proceedings*, pages 359–367, 2012.