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. 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 present a PTAS for H-minor free graphs, if the demands on the edges are bounded above by a constant, and we show that the problem is APX-hard even for cubic graphs and bounded degree bipartite graphs with $c_v = 1$, $\forall v \in V$.

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] in graphs, Maximum Matchings in graphs[10, 12], and the Knapsack Problem [16]. 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 author is supported by a TCS Scholarship

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 [19] even in the case of unit capacities.

Motivated by an application in computing on binary strings by Bu, et al. [6], Zhang introduced the PEPD problem in [19]. 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 [19]).

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.

The rest of the paper is organized as follows. In Section 2, we present the notation used, give a formal definition of the problems studied, and present preliminary results. Section 3 describes related work. We study the PDBEP problem with unit capacities in Section 4, and then present results for the general setting in Section 5. We give a PTAS for the weighted PDBEP problem on H-minor

free graphs in Section 6. In Section 7, 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 $N_e(v) = \{e = \{u, v\} \in E\}$ denote the set of edges incident on v. In the weighted setting, we let $N_{\max}(v) = \{e \in N_e(v) : w(e) \geq w(f)\ \forall f \in N_e(v)\}$. Thus, $N_{\max}(v)$ 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 now formally define the Weighted Partial Degree Bounded Edge Packing Problem with Demands, denoted PEpD: The input is 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 compute a sub-graph $G' = (V, E')$ such that $w(E')$ is maximized, and each edge 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 $e = \{u, v\} \in E'$, $d(N_e(u)) \leq c_u$ or $d(N_e(v)) \leq c_v$. For an edge $e = \{u, v\}$ an end-point that is not overloaded, we call a *good end-point* of e. If neither end-point is overloaded, we pick a good end-point arbitrarily. We also consider the unit-capacity case, i.e., $c_v = 1,\ \forall v \in V$. In this case, we assume $d_e = 1,\ \forall e \in E$. We use PEp to denote the unit-capacity problem.

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 [19].

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[17]. 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 [18], 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 [17], it follows that Or can be solved in polynomial time. □

A similar reduction, imples 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 [15], the lemma follows. □

3 Related Work

The PDBEP problem was introduced by Zhang [19] 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 [11], 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 PEp problem, as stated earlier is similar to the Maximum weight matching problem, for which several polynomial time algorithms are known [10, 13]. The demand version of the problem, PEpD is similar to the *demand matching* problem introduced by Shepherd and Vetta [18]. 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 [15, 14].

The degree-bounded orienting problem Or is a classic combinatorial optimization problem. However, it has mostly received attention in terms of maintaining connectivity. See [17] for more details.

4 A 2-approximation algorithm for Unit Capacity instances

In this section, we present a 2-approximation algorithm for PEp. 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 [19], 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 $\mathrm{OPT}_{\mathrm{PEP}}$ can be constructed from this subset. Recall that for a vertex $v \in V$, $N_{\max}(v)$ denotes the set of edges of maximum weight incident on v, and we use $e_{\max}(v)$ to denote an edge in $N_{\max}(v)$.

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 $N_{\max}(v_i)$ has not been chosen, pick an arbitrary edge $e = \{v_i, v_j\}$ from $N_{\max}(v_i)$. If $e \in N_{\max}(v_j)$ and no edges from $N_{\max}(v_j)$ 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 $N_{\max}(v)$ is chosen for each vertex. This is encoded in the Proposition below.

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

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

Proof. We show that $\sum_{v \in V} w(e_{\max}(v))$ is an upper-bound on the Max-orientation problem Or on the same instance. Then, the lemma follows from Proposition 1. Let $F \subseteq E$ be a feasible solution to the 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. For example consider the graph $G = (\{a,b,c,d\}, \{\{a,b\}, \{b,c\}, \{c,d\}, \{a,d\}, \{b,d\}\})$. Let the weights on the edges be $\{5,3,4,1,2\}$ in the same order. In this example, $\sum_{e \in E_{\max}} w(e) = w(a,b) + w(d,c) = 9$. However, $\mathrm{OPT}_{\mathrm{PEP}} = w(a,b) + w(b,c) + w(b,d) = 10$.

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 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 $e \in N_{\max}(v)$. 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 $N_{max}(v)$ 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 $\{v_k, v_0\} \in N_{\max}(v_k)$, 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 discussed above shows that $w(E_{\max})$ is not an upper bound on OPT_{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. $\text{OPT}_{\text{PEP}} \leq w(E^1_{\max}) + 2w(E^2_{\max})$

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 $N_{\max}(v_i)$ to $E_{\max}$. Let j be the minimal iteration such that an edge from the set $N_{\max}(v_i)$ 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 $N_{\max}(v)$ 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 $\text{OPT}_{\text{PEP}} \leq w(E^1_{\max}) + 2w(E^2_{\max})$. □

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 $\overrightarrow{E}_{\max}$ crosses 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 $\sum_{e \in \overrightarrow{\delta}(X)} w(e) + \sum_{e \in \overrightarrow{\delta}(Y)} w(e) = w(E^1_{\max}) + 2w(E^2_{\max})$, since each edge in $E^2_{\max}$ is present in both $\overrightarrow{\delta}(X)$ and $\overrightarrow{\delta}(Y)$, where for any Z, $\overrightarrow{\delta}(Z)$ denotes the set of out-going edges from Z. Therefore, $\max\{\overrightarrow{\delta}(X), \overrightarrow{\delta}(Y)\} \geq (w(E^1_{\max}) + 2w(E^2_{\max}))/2$. Thus, by returning the maximum among $\overrightarrow{\delta}(X)$, and $\overrightarrow{\delta}(Y)$, we guarantee a solution of weight at least $\mathrm{OPT}_{\text{PEP}}/2$ (Using Lemma 5).

Now it remains to prove the feasibility of $\overrightarrow{\delta}(X)$, and $\overrightarrow{\delta}(Y)$. Note that the in-degree of any vertex is at most 1 in the oriented graph, which ensures both $\overrightarrow{\delta}(X)$ and $\overrightarrow{\delta}(Y)$ 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, 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, there exists a directed multi-graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ with $w(E')$ at least $(1-\epsilon)\mathrm{OPT}_{\text{PEPD}}$ such that $\sum_{e \in in(v)} d_e \leq c_v$, for all $v \in V$.*

Proof. Let $\mathrm{OPT}_{\text{PEPD}}$ 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 $\mathrm{OPT}^i_{\text{PEPD}}$ to be the total weight of the edges incident on v_i in F, otherwise we set $\mathrm{OPT}^i_{\text{PEPD}}$ to be 0. Since for any edge in F, the degree condition is satisfied at at least one end-point, we have $\sum_{i=1}^{n} \mathrm{OPT}^i_{\text{PEPD}} \geq \mathrm{OPT}_{\text{PEPD}}$.

For all $v_i \in V$, we consider the problem of picking a maximum weight sub-set of edges from $N_e(v_i)$ 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 [16], 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

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

The second inequality is true because $\mathrm{OPT}^i_{\text{PEpD}}$ 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 $(1-\epsilon)\mathrm{OPT}_{\text{OrD}}$. 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 $\sum_{e\in E'} w(e) = \sum_{i=1}^{n} w(\mathcal{A}_i) \geq (1-\epsilon)\mathrm{OPT}_{\text{PEpD}}$.

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 $w(E') \geq (1-\epsilon')\mathrm{OPT}_{\text{PEpD}}$, 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 $(1-\epsilon')\mathrm{OPT}_{\text{PEpD}}/2$. Choosing $\epsilon' = \epsilon/(2+\epsilon)$, we obtain a solution of weight at least $\mathrm{OPT}_{\text{PEpD}}/(2+\epsilon)$ 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 $\overrightarrow{\delta}(X)$). 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 $\overrightarrow{\delta}(X)$ 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 $\overrightarrow{\delta}(X)$ directed towards v, the degree condition of v is satisfied in G. This ensures $\overrightarrow{\delta}(X)$ 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

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

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 $\mathrm{OPT}_{\mathrm{PEpD}}$ be an optimal solution to PEpD. Lemma 6 shows that we can obtain an oriented graph $\overrightarrow{G}' = (V, \overrightarrow{E}')$ having $w(E') \geq (1 - \epsilon')\mathrm{OPT}_{\mathrm{PEpD}}$, 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 $(1 - \epsilon')\mathrm{OPT}_{\mathrm{PEpD}}/4$. Using $\epsilon' = \epsilon/(4+\epsilon)$, we obtain a PEpD solution of weight at least $\mathrm{OPT}_{\mathrm{PEpD}}/(4+\epsilon)$.

6 PTAS for PEpD on Minor-free Graphs

In this section, we obtain a PTAS for PEpD in 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 work in the setting where the demands on the edges are bounded by a constant. Then, a PTAS for H-minor-free graphs then follows from the results of Demaine, et al. [9]. For ease of exposition, we only describe the polynomial time algorithm for bounded tree-width graphs with $d_e = 1$ for all $e \in E$. However, the extension to arbitrary, but constant demands is straight forward.

6.1 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 $i \in I$, $Y_i = \{v \in X_j | \ j \text{ is a descendant of } i\}$. 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 bag X_i, let $\mathbf{d_i}$ be the degree sequence of the vertices in X_i in the subgraph $G'_i = (V, F'_i \cup E'_i)$. Suppose $\mathbf{f}$ be a vector representing whether $\deg_{G'}(v) \leq c_v$, or $\deg_{G'}(v) > c_v$. For any $v \in V$, we set $\mathbf{f}(v) = c_v$, if $\deg_{G'}(v) \leq c_v$, 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 $\mathsf{C}(i, E'_i, \mathbf{d_i}, \mathbf{f_i})$ 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 the degrees on vertices of X_i in the subgraphs (V, E') and $(V, F'_i \cup E'_i)$ are bounded above by vectors $\mathbf{f_i}$ and $\mathbf{d_i}$ respectively.

We calculate each DP cell as follows: for each node i, we guess the set of edges E'_i, and the vectors $\mathbf{f_i}$, $\mathbf{d_i} \preceq \mathbf{f_i}$, where $\mathbf{d_i} \preceq \mathbf{f_i}$ denotes that the vector $\mathbf{d_i}$ is component-wise less than or equal to the vector $\mathbf{f_i}$. Since $\mathbf{f}$ provides an upper bound on the degree of any vertex in G', edges in any subgraph of G' must satisfy the feasibility constraint assuming degree of vertex v can be at most $\mathbf{f}(v)$. 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:

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

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:

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

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

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

Where, $\deg_{E'_i \setminus (E'_j \cup E'_k)}(v)$ and $\deg_{E'_j \cap E'_k}(v)$ denote the degrees of 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 $\max\{\mathsf{W}(r, E'_r, \mathbf{f_r}, \mathbf{f_r})\}$, 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 $i \in I$, $|X_i| \leq t$, 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 n values. $\mathbf{d_i}(v)$ can take at most n values. So, total number of cells in the DP can be at most $O(nt) \cdot 2^{t^2} \cdot 2^t \cdot n^t = O(n^t)$. Each DP cell takes $O(n^{2t})$ computation time. So the running time of the DP is bounded by $O(n^{3t})$.

Theorem 4. *The* PEPD *problem, on graphs with bounded tree-width can be solved in polynomial time if the demand of any edge is bounded above by a constant.*

6.2 Partition into Bounded Treewidth Graphs

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

Lemma 9 ([9]). *For a fixed graph H, there is a constant c_H such that, for any integer $k \geq 1$ 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 5. *In H-minor free graphs, there is a PTAS for the* PEPD *problem if the demand of any edge is bounded above by a constant.*

Proof. Let $G = (V, E)$ be any H-minor-free graph. We apply lemma 9 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 4. Since the set $E' \setminus E'_i$ is a feasible solution to G_i, we have $OPT_i \geq w(E' \setminus E'_i)$.

$$\begin{aligned} \max\{OPT_1, \cdots, OPT_{k+1}\} \geq OPT_m &\geq w(E' \setminus E'_m) \\ &\geq \left(1 - \frac{1}{k+1}\right) w(E') \geq (1 - \epsilon) w(E'). \end{aligned}$$

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

7 APX-hardness

In this section, we prove that PEp problem is APX-hard even for unweighted cubic graphs, and unweighted bipartite graphs of bounded degree. Earlier, only NP-hardness was known. This was proved by Zhang [19] 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: Cubic graphs and bipartite graphs of bounded degree have large dominating set, and the fact that the Dominating set problem is APX-hard on cubic graphs and bounded degree bipartite graphs. We show that a PTAS for the unweighted PEp would imply a PTAS for the Dominating set problem.

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

Proof. Let $|U \cup V| = n$ and suppose there is a dominating set $S \subset U \cup V$ of size $< n/(1+B)$. Since each $v \in S$ can dominate at most B vertices in $\{U \cup V\} \setminus S$, all vertices in S together can dominate $< nB/(1 + B)$ 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 $G < n/(1 + B) + nB/(1 + B) = n$ contradicting our assumption that $|U \cup V| = n$.

We use the following result of Zhang [19] on the relation between Dominating Sets and PEp on unweighted graphs. Following which, our result on bipartite graphs follows by using the result of Chlebík and Chlebíková [7] that Dominating Set on bounded degree bipartite graphs is APX-hard.

Lemma 10 ([19]). *Let $G = (V, E)$ be a graph without isolated vertices having $w(e) = 1$, for all $e \in E$. In G, there is a maximal solution of size k to the* PEp *if and only if there is a solution of size $|V| - k$ to the Dominating Set problem.*

Theorem 6 ([7]). *It is NP-hard to approximate the Dominating Set problem in bipartite graphs of degree at most $B \geq 3$ within a factor of $\ln B - C \ln \ln B$ for some absolute constant C.*

Theorem 7. *The* PEP *problem is APX-hard for unweighted bipartite graphs having degree at most $B \geq 3$.*

Proof. We prove that an existence of a PTAS for the PEP problem on bounded degree bipartite graphs implies a PTAS for the Dominating Set problem on the same class of graphs, contradicting Theorem 6.

Let $G = (U \cup V, E)$ be a bipartite graph with degree bounded by B and $|U \cup V| = n$. By Proposition 3, $\mathrm{OPT}_{DS}(G) \geq n/(1+B)$. Lemma 10 implies that $\mathrm{OPT}_{\mathrm{PEP}}(G) = n - \mathrm{OPT}_{DS}(G)$. If there exists a PTAS for PEP, this implies that for every $\epsilon > 0$, we can find a sub-graph $G' = (V, E')$ such that $|E'| \geq (1-\epsilon)(n - \mathrm{OPT}_{DS}(G))$. We can assume that G' is a collection of disjoint stars with no isolated vertices as a PEP solution with isolated vertices can always be transformed into one without isolated vertices. Therefore G' is a collection of stars spanning V and the set C of central vertices form a dominating set (Lemma 10). $|C| = n - |E'|$. Therefore,

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

Therefore, PEP is APX-hard, even on unweighted bipartite graphs of bounded degree.

Remark: The APX-hardness result on cubic graphs can be obtained by using a similar argument and the result of Alimonti and Kann [1] that Dominating Set on cubic graphs is APX-hard.

8 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 $\mathrm{OPT}_{\mathrm{PEP}}$, $\mathrm{OPT}_{\mathrm{PEPD}}$ on bipartite graphs, and $\mathrm{OPT}_{\mathrm{PEPD}}$ on general graphs respectively. For example consider the graph C_4 (cycle on 4 vertices) with unit weight on all the edges. In this case $\mathrm{OPT}_{\mathrm{PEP}} = 2$, whereas the upper bound is $\sum_{v \in V} w(e_{\max}(v)) = 4$. So we cannot expect to get a better than 2-approximation with this upper bound.

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 with arbitrary bounds. In *Frontiers in Algorithmics* and *Algorithmic Aspects in Information and Management, Third Joint International Conference, FAW-AAIM 2013, Dalian, China, June 26-28, 2013. Proceedings*, pages 24–35, 2013.
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. Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008.
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. Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.
12. László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.
13. 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.
14. Ojas Parekh. Iterative packing for demand and hypergraph matching. In *Integer Programming and Combinatoral Optimization*, pages 349–361. Springer, 2011.
15. Ojas Parekh and David Pritchard. Generalized hypergraph matching via iterated packing and local ratio. In *Approximation and Online Algorithms*, pages 207–223. Springer, 2014.
16. Sartaj Kumar Sahni. On the knapsack and other computationally related problems. 1973.
17. Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
18. 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.
19. 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.