Partial Degree Bounded Edge Packing Problem with Arbitrary Bounds

Pawan Aurora, Sumit Singh, and Shashank K Mehta

Indian Institute of Technology, Kanpur - 208016, India
paurora@iitk.ac.in,ssumit@iitk.ac.in,skmehta@cse.iitk.ac.in

Abstract. Given a graph $G = (V, E)$ and a non-negative integer c_u for each $u \in V$. Partial Degree Bounded Edge Packing (PDBEP) problem is to find a subgraph $G' = (V, E')$ with maximum $|E'|$ such that for each edge $(u, v) \in E'$, either $deg_{G'}(u) \leq c_u$ or $deg_{G'}(v) \leq c_v$. The problem has been shown to be NP-hard even for uniform degree constraint (i.e., all c_u being equal). Approximation algorithms for uniform degree cases with c_u equal to 1 and 2 with approximation ratio of 2 and 32/11 respectively are known. In this work we study general degree constraint case (arbitrary degree constraint for each vertex) and present two combinatorial approximation algorithms with approximation factors 4 and 2. We also study a related integer program for which we present an iterative rounding algorithm with approximation factor $1.5/(1-\epsilon)$ for any positive ϵ. This also leads to a $3/(1-\epsilon)^2$ factor approximation algorithm for the general PDBEP problem. For special cases (large values of c_v/deg_v's) the factor improves up to $1.5/(1-\epsilon)$. Next we study the same problem with weighted edges. In this case we present a $2 + \log_2 n$ approximation algorithm. In the literature exact $O(n^2)$ complexity algorithm for trees is known in case of uniform degree constraint. We improve this result by giving an $O(n \cdot \log n)$ complexity exact algorithm for trees with general degree constraint.

Keywords: Edge-Packing Problems, Iterative Rounding, Lagrangian Relaxation.

1 Introduction

The *partial degree bounded edge packing problem* (PDBEP) is described as follows: Given a graph $G = (V, E)$ and degree-bound function $c : V \to \mathbb{Z}^*$ ($\mathbb{Z}^*$ is the set of non-negative integers), compute a maximum cardinality set $E' \subseteq E$ which satisfies the *degree condition*: $(d'_u \leq c_u) \vee (d'_v \leq c_v)$ for each $e = (u, v) \in E'$. Here d'_x denotes the degree of vertex x in the graph $G' = (V, E')$. Without loss of generality, we will assume that $c_v \leq d_v$ for all $v \in V$ where d_v denotes the degree of v in G.

In the weighted version of the problem edges are assigned non-negative weights and we want to compute a set of edges E' with maximum cumulative weight subject to the degree condition described above.

In [4], the PDBEP problem was motivated by an application in binary string representation. It was shown there that the maximum expressible independent subset (MEIS) problem on 2-regular set can be reduced to PDBEP problem with uniform constraint $c = 2$. The PDBEP problem finds another interesting application in resource allocation. Given n types of resources and m jobs, each job needs two types of resources. A job j, which requires resources u and v, can be accomplished if u is not shared by more than c_u jobs or v is not shared by more than c_v jobs. Interpreting the resources as the vertices of the input graph and the jobs as edges, the PDBEP problem is to compute the maximum number of jobs that can be accomplished.

1.1 Related Work

The decision problem of edge packing when there is a uniform degree constraint of 1 is a parametric dual of the Dominating Set (DS) problem. It was studied in [3]. It was also studied under the framework of parameterized complexity by Dehne, Fellows, Fernau, Prieto and Rosamond in [1].

Recently Peng Zhang [4] showed that the PDBEP problem with uniform degree constraint ($c_v = k$ for all v) is NP-hard even for $k = 1$ for general graphs. They gave approximation algorithms for the PDBEP problem under uniform degree constraints of $k = 1$ and $k = 2$ with approximation factors 2 and 32/11 respectively. They showed that PDBEP on trees with uniform degree constraint can be solved exactly in $O(n^2)$ time.

1.2 Our Contribution

We propose three different approximation algorithms for the problem with general degree constraints (i.e., for arbitrary non-negative function c). Two of these algorithms are combinatorial in nature and their approximation ratios are 4 and 2. The third algorithm is a consequence of studying a related integer program (IP) for which we present a $1.5/(1-\epsilon)$ approximation iterative rounding [2] algorithm. It turns out that any α approximation of this IP is a $2\alpha/(1-\epsilon)$ approximation of the PDBEP problem for any $\epsilon > 0$. That gives us a $3/(1-\epsilon)^2$ factor approximation to the PDBEP problem. However for large degree constraint with respect to the degree, the approximation factor can improve up to $1.5/(1-\epsilon)$. The results detailed above are for general graphs with arbitrary degree constraint and the approximation factor is also improved for $c_v = 2$ (uniform constraint) case in [4].

Next we consider the PDBEP problem with arbitrary degree constraint for edge-weighted graphs. In this case we present a combinatorial approximation algorithm with approximation factor of $2+\log_2 n$. Edge-weighted PDBEP problem is not addressed in the literature, to the best of our knowledge.

Finally we present an exact algorithm for unweighted trees with arbitrary degree constraint function. The time complexity of this algorithm is $O(n \log n)$. This is an improvement over the $O(n^2)$ algorithm in [4] which is applicable to only the uniform degree constraint case.

2 Approximation Algorithms for the unweighted case

The optimum solution of a PDBEP problem can be bounded as follows.

Lemma 1. *Let $G = (V, E)$ be a graph with degree-bound function $c : V \to \mathbb{Z}^*$. Then the optimal solution of PDBEP can have at most $\sum_{v \in V} c_v$ edges.*

Proof. Let $E' \subset E$ be a solution of PDBEP. Let $U = \{v \in V | d'_v \leq c_v\}$. Then from the degree condition we see that U is a vertex cover in the graph (V, E'). Hence $|E'| \leq \sum_{u \in U} c_u \leq \sum_{v \in V} c_v$. □

2.1 Edge Addition based Algorithm

Compute a maximal solution Y by iteratively adding edges, i.e., in each iteration select a new edge and add it to Y if it does not result into degree violation on both end-vertices. Let $d_Y(x)$ denote the degree of a vertex x in the graph (V, Y). Partition the vertex set into sets: $A = \{v | d_Y(v) < c_v\}$, $B = \{v | d_Y(v) = c_v\}$, and $C = \{v | d_Y(v) > c_v\}$. Observe that every edge of the set $E \setminus Y$ which is incident on a vertex in A, has its other vertex in B. Hence for any $a_1, a_2 \in A$ the $E \setminus E'$ edges incident on a_1 are all distinct from those incident on a_2. Construct another edge set Z containing any $c_v - d_Y(v)$ edges from $E \setminus Y$, incident on v for each $v \in A$. Observe that Z also satisfies the degree constraints. Output the larger of Y and Z. See Algorithm 1. We have the following result about the correctness of the algorithm.

Lemma 2. *The algorithm outputs a set which satisfies the degree constraint.*

Consider the set $Y \cup Z$. In this set the degree of each vertex is not less than its degree-bound. Hence the cardinality of the output of the algorithm is at least $\sum_v c_v/4$. From Lemma 1 the approximation ratio is bounded by 4.

Theorem 1. *The Algorithm 1 has approximation factor* 4.

2.2 Edge Deletion based Algorithm

The second algorithm for PDBEP is based on elimination of edges from the edge set. Starting with the input edge set E, iteratively we delete the edges in violation, i.e., in each iteration one edge (u, v) is deleted if the current degree of u is greater than c_u and that of v is greater than c_v. The surviving edge set Y is the result of the algorithm. See Algorithm 2.

Clearly Y satisfies the degree condition. Also observe that $d_Y(v) \geq c_v$ for all $v \in V$. Hence $|Y| \geq \sum_v c_v/2$. From Lemma 1, $|Y| \geq OPT/2$ where OPT denotes the optimum solution.

Theorem 2. *The Algorithm 2 has approximation ratio* 2.

Data: A connected graph $G = (V, E)$ and a function $c : V \to \mathbb{Z}^*$ such that $c_v \leq d(v)$ for each vertex v.
Result: Approximation for the largest subset of E which satisfies the degree-condition.

```
Y := ∅;
for e ∈ E do
    if Y ∪ {e} satisfies the degree-condition then
        Y := Y ∪ {e};
    end
end
Compute A := {v ∈ V | d_Y(v) < c_v};
Z := ∅;
for v ∈ A do
    Select arbitrary c_v − d_Y(v) edges incident on v in E \ Y and insert into Z;
end
if |Y| ≥ |Z| then
    return Y;
else
    return Z;
end
```

Algorithm 1: Edge Addition Based Algorithm

2.3 LP based Algorithm

In this section we explore a linear programming based approach to design an approximation algorithm for PDBEP.

The Integer Program The natural IP formulation of the problem is as follows.

$$\text{IP1: max } \psi = \sum_{e \in E} y_e, \text{ subject to}$$
$$y_e \leq x_u + x_v \ \forall e = (u, v) \in E,$$
$$\sum_{e \in \delta(v)} y_e \leq c_v x_v + d_v(1 - x_v) \ \forall v \in V,$$
$$\text{where } \delta(v) \text{ denotes the set of edges incident on } v,$$
$$x_v \in \{0, 1\} \ \forall v \in V,\ y_e \in \{0, 1\} \ \forall e \in E$$

The solution computed by the program is $E' = \{e | y_e = 1\}$. The linear programming relaxation of the above integer program will be referred to as LP1.

Lemma 3. *The integrality gap of LP1 is $\Omega(n)$ where n is the number of vertices in the graph.*

Proof. Consider the following instance of the problem. Let G be a complete graph on n vertices $\{v_0, v_1, \ldots, v_{n-1}\}$ and the degree constraint be $c_v = 1 \, \forall v \in V$. We now construct a feasible fractional solution of $LP1$ as follows. Let $x_v = 0.5$ for all

Data: A connected graph $G = (V, E)$ and a function $c : V \to \mathbb{Z}^*$ such that c_v is the degree bound for vertex v.
Result: Approximation for the largest subset of E which satisfies the degree-condition.

```
Y := E;
for e = (u, v) ∈ Y do
    if d_Y(u) > c_u and d_Y(v) > c_v then
        Y ← Y \ {e};
    end
end
return Y;
```

Algorithm 2: Edge Deletion Based Algorithm

v and $y_e = 1$ for all $e = (v_i, v_j)$ where j is in the interval $((i - \lfloor n/4 \rfloor) (\text{mod } n), (i + \lfloor n/4 \rfloor) (\text{mod } n))$. The value of the objective function for this solution is at least $(n-1)^2/4$. On the other hand, from Lemma 1, the optimal solution for the IP1 cannot be more than n. Hence the integrality gap is $\Omega(n)$. □

High integrality gap necessitates an alternative approach.

Approximate Integer Program We propose an alternative integer program IP2, for any $\epsilon > 0$, which is a form of Lagrangian relaxation of IP1. We will show that its *maximal* solutions are also solutions of IP1 and any α approximation of IP2 is a $2\alpha/(1-\epsilon)$ approximation of IP1. A maximal solution of IP2 is a solution in which changing the value of any y_e or any z_v either renders the solution infeasible or does not improve its objective function value. It is easy to show that in a maximal solution $z_v = \max\{0, \sum_{e \in \delta(v)} y_e - c_v\}$ for all v.

$$\text{IP2: max } \phi = 2 \sum_{e \in E} y_e - (1 + \epsilon) \sum_{v \in V} z_v, \text{ subject to}$$

$$\sum_{e \in \delta(v)} y_e \leq c_v + z_v \; \forall v \in V,$$

$$z_v \in \{0, 1, 2, \ldots\} \; \forall v \in V, \; y_e \in \{0, 1\} \; \forall e \in E$$

Note that any subset of edges E' is a feasible solution of IP2 if we choose $z_v = \max\{0, \sum_{e \in \delta(v)} y_e - c_v\}$ for all v. Besides, these values of z will give maximum value of the objective function over other consistent values of z. Hence z values are not required to be specified in the solutions of IP2. We will denote $\sum_v z_v$ by Z.

Lemma 4. *Every maximal solution of the integer program IP2 is also a feasible solution of PDBEP.*

Proof. Consider any maximal solution E' of IP2. In a maximal solution $z_v = \max\{0, \sum_{e \in \delta(v)} y_e - c_v\}$ for all v. Assume that it is not a feasible solution of

PDBEP. Then there must exist an edge $e = (u, v) \in E'$ such that $z_u \geq 1$ and $z_v \geq 1$. Define an alternative solution $E'' = E' \setminus \{e\}$ and decrement z_u and z_v by 1 each. Observe that the objective function of the new solution increases by 2ϵ. This contradicts that E' is a maximal solution. □

In a maximal solution E' of IP2, Z is the count of excess edges incident on violating vertices. Hence $Z \leq |E'|$. In this case it is easy to see that $Z/|E'| \leq \max_v\{(d_v - c_v)/d_v\}$.

Lemma 5. *Any α approximate solution of IP2, which is also maximal, is a $2\alpha/(1-\epsilon)$ approximation of PDBEP problem.*

Proof. Let E'_1 be an α-approximation maximal solution of IP2 and E'_2 be an optimum solution of PDBEP. Then E'_2 is also a solution of IP2 with $y_{2e} = 1$ for $e \in E'_2$ and $z_{2v} = \max\{0, \sum_{e\in\delta(v)} y_{2e} - c_v\}$. Then $\phi(E'_1) = 2|E'_1| - (1+\epsilon)Z_1$ and $\phi(E'_2) = 2|E'_2| - (1+\epsilon)Z_2$. Let OPT denote the optimum value of IP2. Then $\phi(E'_2) \leq OPT$ and $OPT/\alpha \leq \phi(E'_1)$. So $2|E'_2| - (1+\epsilon)Z_2 \leq \alpha(2|E'_1| - (1+\epsilon)Z_1)$. Suppose $Z_2 \leq \beta|E'_2|$. Then $|E'_2|/|E'_1| \leq 2\alpha/(2 - (1+\epsilon)\beta)$. Since $\beta \leq 1$, we get the desired approximation factor. □

Corollary 1. *If $c_v \geq (1-\beta)d_v \, \forall v$, then any α approximate solution of IP2, which is also maximal, is a $2\alpha/(2-(1+\epsilon)\beta)$ approximation of PDBEP.*

In this case $Z_2 \leq \beta|E'_2|$. Now the result follows from the previous proof.

2.4 Algorithm for IP2

We propose Algorithm 3 which approximates the IP2 problem within a constant factor of approximation. LP2 is the linear program relaxation of IP2. Here we assume that an additional constraint is imposed, namely, $\{z_v = 0 | v \in C\}$ where we require a solution in which every $v \in C$ must necessarily satisfy the degree constraint. The input to the problem is $(H = (V, E), C)$. Algorithm starts with $E' = \emptyset$ and builds it up one edge at a time by iterative rounding. In each iteration we discard at least one edge from further consideration. Hence it requires at most $|E|$ iterations (actually it requires at most $|V|+1$ iterations, see the remark below.) To simplify the analysis Algorithm 3 is presented in the recursive format.

$$\text{LP2: max } \phi = 2\sum_{e\in E} y_e - (1+\epsilon)\sum_{v\in V} z_v, \text{ subject to}$$

$$\sum_{e\in\delta(v)} y_e \leq c_v + z_v \; \forall v \in V \setminus C,$$

$$\sum_{e\in\delta(v)} y_e \leq c_v \; \forall v \in C,$$

$$z_v \geq 0 \; \forall v \in V, \; y_e \geq 0 \; \forall e \in E, \; -y_e \geq -1 \; \forall e \in E$$

In the following analysis we will focus on two problems: (H, C) of some i-th nested recursive call and (H_1, C_1) of the next call in Algorithm 3. For simplicity we will refer to them as the problems of graphs H and H_1 respectively.

Lemma 6. *In a corner solution of LP2 on a non-empty graph there is at least one edge e with $y_e = 0$ or $y_e \geq 1/2$.*

Proof. Assume the contrary that in an extreme point solution of LP2 all y_e are in the open interval $(0, 1/2)$. Let us partition the vertices as follows. Let n_1 vertices have $c_v > 0$ and $z_v > 0$, n_2 vertices have $c_v > 0$ and $z_v = 0$ and n_3 vertices have $c_v = 0$ and $z_v > 0$. Note that the case of $c_v = 0$ and $z_v = 0$ cannot arise because $y_e > 0$ for all e. In each case let n_i' vertices have the condition $\sum_{e \in \delta(v)} y_e \leq c_v + z_v$ tight (an equality) and n_i'' vertices have the condition a strict inequality. Let the number of edges be m.

The total number of variables is $n_1+n_2+n_3+m$. In $n_1'+n_2'$ cases $\sum_{e \in \delta(v)} y_e = c_v + z_v$ where $c_v \geq 1$ and each $y_e < 0.5$ so there must be at least 3 edges incident on such vertices. Since the graph has no isolated vertices, every vertex has at least one incident edge. Hence $m \geq (3n_1' + 3n_2' + n_1'' + n_2'' + n_3)/2$. So the number of variables is at least $n_1' + n_2' + (1.5)(n_1 + n_2 + n_3)$.

Now we find the number of tight conditions. None of the y_e touches their bounds. The number of z_v which are equal to zero is n_2, and the number of instances when $\sum_{e \in \delta(v)} y_e = c_v + z_v$ is $n_1' + n_2' + n_3'$. Hence the total number of conditions which are tight is $n_2 + n_1' + n_2' + n_3'$. Since the solution is an extreme point, the number of tight conditions must not be less than the number of variables. So $n_2 + n_1' + n_2' + n_3' \geq n_1' + n_2' + (1.5)(n_1 + n_2 + n_3)$. This implies that $n_1 = n_2 = n_3 = 0$, which is absurd since the input graph is not empty. □

Remark: The program LP2 has $|E|+|V|$ variables and $2|E|+2|V|$ constraints. Hence in the first iteration the optimal solution must have at least $|E| - |V|$ tight edge-constraints (i.e., $y_e = 0$ or $y_e = 1$.) All these can be processed simultaneously so in the second round at most $|V|$ edges will remain in the residual graph. Thus the total number of iterations cannot exceed $|V| + 1$.

Lemma 7. *If $y_e \geq 1/2$ in the solution of LP2 where $e = (u, v)$, then $(c_u > 0, z_u = 0)$ or $(c_v > 0, z_v = 0)$.*

Proof. Assume that $z_v > 0$ and $z_u > 0$ in the solution. Let the minimum of z_v, z_u, and y_e be β. Subtracting β from these variables results in a feasible solution with objective function value greater than the optimum by $2\beta \cdot \epsilon$. This is absurd. Hence z_u and z_v both cannot be positive.

Next assume that $c_u = 0$ and $z_u = 0$. Then y_e must be zero, contradicting the fact that $y_e \geq 1/2$. Similarly $c_v = 0$ and $z_v = 0$ is also not possible.

Therefore at least one of $(c_u > 0, z_u = 0)$ and $(c_v > 0, z_v = 0)$ holds. □

Lemma 8. *The Algorithm 3 returns a feasible solution of PDBEP.*

Proof. The claim is trivially true when the graph is empty. We will use induction.

In the case of $y_e = 0$, the solution of H_1 is also a solution of H. From induction hypothesis it is feasible for PDBEP for H_1 hence it is also feasible for PDBEP for H.

Consider the second case, i.e., $y_e \geq 1/2$. Let $e = (u, v)$. From Lemma 7 $f_v = a > 0$ and $z_v = 0$. Since $z_{1v} = 0$ and $f_{1v} = a - 1$, in the solution of H_1 at

most $a-1$ edges can be incident on v. So in the solution of H, there are at most a edges incident on v and $f_v = a$. Thus e is valid in the solution of H.

Now consider vertex u in this case. If $f_u > 0$, then it is similar to v. But if $f_u = 0$, then f_{1u} is also 0. If $z_{1u} > 0$, i.e., the solution of H_1 has u violating, then edges incident on u must have their other ends on non-violating vertices. Replacing e will not affect any other edges. Next if $z_{1u} = 0$ then solution of H_1 will have no edge incident on u. Now putting back e, we find e is incident on u and $f_u = 0$ so u turns into a violating vertex ($z_u = 1$). But e is the only edge incident on u and its other end v is not violating. Other edges are valid due to induction hypothesis. Hence the solution of H is a feasible solution of PDBEP. □

Now we analyze the performance of the algorithm.

Data: A connected graph $G = (V, E)$ and a function $c : V \to \mathbb{N}$
Result: A solution of PDBEP problem.
for $v \in V$ **do**
  $f_v := c_v$;
end
$C := \emptyset$;
$E' := SolveIP2(G, C, f)$; `/* see the function SolveIP2 */`
return E';

Algorithm 3: Iterative Rounding based Algorithm in Recursive Format

Lemma 9. *Algorithm 3 gives a* $1.5/(1-\epsilon)$ *approximation of IP2.*

Proof. Let c denote $1.5/(1-\epsilon)$. We will denote the optimal LP2 solutions of H and H_1 by F and F_1 respectively. Similarly I and I_1 will denote the solutions computed by the algorithm for H and H_1 respectively. f_{1*} and z_{1*} denote the parameters associated with H_1. We will assume that $z_x = \max\{0, \sum_{e\in\delta(x)} y_e - f_x\}$ for integral solutions to compute their ϕ-values. Again we will prove the claim by induction. The base case is trivially true. From induction hypothesis $\phi(F_1)/\phi(I_1) \leq c$ and our goal is to show the same bound holds for $\phi(F)/\phi(I)$.

In the event of $y_e = 0$ in F, $\phi(F) = \phi(F_1)$ and $\phi(I) = \phi(I_1)$. Hence $\phi(F)/\phi(I) = \phi(F_1)/\phi(I_1)$.

In case $y_e = \alpha \geq 1/2$ we will consider two cases: (i) $f_u > 0$ and (ii) $f_u = 0$ in F. In the first case I differs from I_1 in three aspects: $y_e = 1$ in I, $f_v = f_{1v} + 1$ and $f_u = f_{1u} + 1$. So z_v and z_u remain unchanged, i.e., $z_v = z_{1v}$ and $z_u = z_{1u}$. Thus $\phi(I) = \phi(I_1) + 2$. In the second case also y_e increases by 1 and z_v remains unchanged but z_u increases by 1 because in this case $f_u = f_{1u} = 0$. Hence $\phi(I) = \phi(I_1) + 1 - \epsilon$.

In the remaining part of the proof we will construct a solution of LP2 for H_1 from F, the optimal solution of LP2 for H.

Again we will consider the two cases separately. First the case of $f_u > 0$. Set $y_e = 0$. If $\sum_{e'\in\delta(v)\setminus\{e\}} y_{e'} \geq 1-\alpha$ then subtract the values of $y_{e'}$ for $e' \in \delta(v)\setminus\{e\}$ in arbitrary manner so that the sum $\sum_{e'\in\delta(v)\setminus\{e\}} y_{e'}$ decreases by $1-\alpha$. If $\sum_{e'\in\delta(v)\setminus\{e\}} y_{e'} < 1-\alpha$, then set $y_{e'}$ to zero for all edges incident on v. Repeat

```
Function: SolveIP2(H = (V_H, E_H), C, f)
if E_H := ∅ then
  | return ∅;
end
Delete all isolated vertices from V_H;
(y, z) = LPSolver(H, C);
/* solve LP2 with degree-bounds f(x) for all x ∈ V_H                    */
if ∃e ∈ E_H with y_e = 0 then
  | H_1 := (V_H, E_H \ {e});
  | C_1 := C;
  | E' := SolveIP2(H_1, C_1, f);
else
  | From Lemma 6 there exists an edge e := (u, v) with y_e ≥ 1/2;
  | From Lemma 7 without loss of generality we assume (f_v > 0, z_v = 0);
  | f_v := f_v − 1;
  | C_1 := C ∪ {v};
  | f_u := max{f_u − 1, 0};
  | H_1 := (V_H, E_H \ {e});
  | E' := SolveIP2(H_1, C_1, f) ∪ {e};
  | /* Including e in E' means y_e is rounded up to 1. In case f_u = 0,
  |    z_u is implicitly raised to ensure that Σ_{e'∈δ(u)} y_e' ≤ f_u + z_u
  |    continues to hold. We do not explicitly increase z_u value
  |    since it is not output as a part of the solution.               */
end
return E';
```

this step for edges incident on u. Retain values of all other variables as in F (in particular, the values of z_u and z_v). Observe that these values constitute a solution of LP2 for H_1. Call this solution F_1'. From induction hypothesis $\phi(F_1) \leq c\phi(I_1)$. Hence we have $\phi(F_1') \leq c\phi(I_1)$. Then $\phi(F_1') \geq \phi(F) - 2(1 + 1 - \alpha) \geq \phi(F) - 3$. We have $\phi(F) \leq \phi(F_1') + 3 \leq c\phi(I_1) + 3 = c(\phi(I) - 2) + 3 \leq c\phi(I)$.

In the second case $f_u = 0$. Once again repeat the step described for edges incident on v and set y_e to zero. In this case $z_u \geq \alpha$ so subtract α from it. It is easy to see that again the resulting variable values form an LP2 solution of H_1, call it F_1'. So $\phi(F_1') = \phi(F) - (2 - (1 + \epsilon)\alpha)$. So $\phi(F) = \phi(F_1') + (2 - (1 + \epsilon)\alpha) \leq c\phi(I_1) + (2 - (1 + \epsilon)\alpha)$. Plugging $\phi(I) - 1 + \epsilon$ for $\phi(I_1)$ and simplifying the expression gives $\phi(F) \leq c\phi(I)$. This completes the proof. □

Combining lemmas 5 and 9 we have the following result.

Theorem 3. *Algorithm 3 approximates PDBEP with approximation factor* $3/(1-\epsilon)^2$.

From Corollary 1 we have the following result.

Corollary 2. *If* $c_v \geq (1-\beta)d_v$ *for all* v*, then Algorithm 3 approximates PDBEP with approximation factor* $3/((2 - (1 + \epsilon)\beta)(1 - \epsilon))$.

3 Approximation Algorithm for the weighted case

Let $H(v)$ denote the heaviest c_v edges incident on vertex v, called *heavy set* of vertex v. Then from a generalization of Lemma 1 the optimum solution of PDBEP in weighted-edge case is bounded by $\sum_{v \in V} \sum_{e \in H(v)} w(e)$ where $w(e)$ denotes the weight of edge e. We will describe a method to construct upto $1 + \log |V|$ solutions, which cover $\cup_{v \in V} H(v)$. Then the heaviest solution gives a $2 + \log |V|$ approximation of the problem.

3.1 The Algorithm

Input: A graph (V, E) with non-negative edge-weight function $w()$. Let $|V| = n$.

Step 0: Add infinitesimally small weights to ensure that all weights are distinct, without affecting heavy sets.

Step 1: $E_1 = E \setminus \{e = (u, v) \in E | e \notin H(u) \text{ and } e \notin H(v)\}$.

Step 2: $T = \{e = (u, v) \in E | e \in H(u) \text{ and } e \in H(v)\}$.

Step 3: $E_2 = E_1 \setminus T$. Clearly each edge of E_2 is in the heavy set of only one of its end-vertices. Suppose $e = (u, v) \in E_2$ with $e \notin H(u)$ and $e \in H(v)$. Then we will think of e as directed from u to v.

Step 4: Label the vertices from 0 to $n - 1$ such that if edge (u, v) is directed from u to v, then $Label(u) < Label(v)$. Define subsets of E_2-edges, $A_0, \ldots, A_{k-1}$, where $k = \log_2 n$, as follows. A_r consists of edges (u, v) directed from u to v, such that the most significant $r - 1$ bits of binary expansion of the labels of u and v are same and r-th bit differs. Note that this bit will be 0 for u.

Step 5: Output that set among the $\log n + 1$ sets, $T, A_0, \ldots, A_{k-1}$, which has maximum cumulative edge weight.

Theorem 4. *The algorithm gives a feasible solution with approximation factor* $2 + \log_2 n$.

Proof. Set T constitutes a feasible solution since both ends of each edge in it satisfy the degree constraint. The directed E_2 edges define an acyclic graph, hence the labeling can be performed by topological sorting. Clearly $E_2 = \cup_r A_r$. In A_r all arrows are pointed from u with r-th most significant bit zero to v with r-th most significant bit one. Hence it is a bipartite graph where all arrows have heads in one set and the tails in the other. All vertices on the head side satisfy the degree conditions because all their incident edges are in their heavy sets. Therefore A_r are feasible solutions. We have $T \cup (\cup_r A_r) = E_1$. Observe that $\cup_v H(v) = E_1$. Only T-edges have both ends in heavy sets. Using the fact that $OPT \leq \sum_v w(H(v))$, we deduce that $OPT \leq 2w(T) + \sum_r w(A_r)$. So the weight of the set output in step 5 is at least $OPT/(2 + \log_2 n)$. □

4 Exact Algorithm for Trees

In this section we give a polynomial time exact algorithm for the unweighted PDBEP problem for the special case when the input graph is a tree. We will

denote the degree of a vertex v in the input graph by $d(v)$ and its degree in a solution under consideration by $d'(v)$.

Let T be a rooted tree with root R. For any vertex v we denote the subtree rooted at v by $T(v)$. Consider all feasible solutions of PDBEP of graph $T(v)$ in which degree of v is at most $c_v - 1$, call them H-solutions (white). Let $h(v)$ be the number of edges in the largest such solution. Similarly let $g(v)$ be the optimal G-solution (grey) in which the degree of v is restricted to be equal to c_v. Lastly $b(v)$ will denote the optimal B-solution (black) which are solutions of $T(v)$ under the restriction that degree of v be at least c_v and every neighbor of v in the solution satisfies the degree condition. It may be observed that one class of solutions of $T(v)$ are included in G-solutions as well as in B-solutions. These are the solutions in which $d'(v) = c_v$ and every neighbor u of v in the solution has $d'(u) \leq c_u$. If in any of these cases there are no feasible solutions, then the corresponding optimal value is assumed to be zero. Hence the optimum solution of PDBEP for T is the maximum of $h(R), g(R)$, and $b(R)$ and all three values are zero for leaf nodes.

Let $Ch(v)$ denote the set of child-nodes of v in $T(v)$. We partition $Ch(v)$ into $H(v) = \{u \in Ch(v) | h(u) \geq \max\{g(u), b(u)\}\}$, $G(v) = \{u \in Ch(v) | g(u) > \max\{h(u), b(u)\}\}$, $B(v) = Ch(V) \setminus (G(v) \cup H(v))$. While constructing a G-solution of $T(v)$ from the solutions of the children of v we can include the edge (v, u) for any vertex u in $H(v) \cup B(v)$ along with the optimal solution of $T(u)$ without disturbing the degree conditions of the edges in this solution. But we can add edge (v, u) to the solution, for any $u \in G(v)$, only by selecting a B-solution or an H-solution of $T(u)$ because if we use a G-solution for $T(u)$, then vertex u which was earlier satisfying the degree condition, will now have degree $c_u + 1$.

Next, while constructing a B-solution of $T(v)$ we can connect v to any number of $H(v)$ vertices and use their optimal H-solutions. In the same case, in order to connect v with $u \in B(v) \cup G(v)$ we must use the optimal H-solution of $T(u)$ (which is not the best solution of $T(u)$).

Suppose we want to build the optimum G-solution of v. If $|H(v)| + |B(v)|$ is less than c_v, then we must pick additional $c_v - |H(v)| - |B(v)|$ vertices from $G(v)$ to connect with v. If $k = c_v - |H(v)| - |B(v)| > 0$, then we define $S'(v)$ to be the set of k members of $G(v)$ having least $g(u) - \max\{h(u), b(u)\}$. Otherwise $S'(v) = \emptyset$. $S'(v)$ are those vertices which we will like to connect v with.

Suppose we want to build the best B-solution for $T(v)$ and $|H(v)| < c_v$. Then we will connect v with exactly c_v children because connecting with any additional child will either keep the value same or make it worse because for connecting with a $G(v)$ or $B(v)$ node we will be forced to use their H-solution which is not their best solution. If $k = c_v - |H(v)| > 0$, then we define $S''(v)$ to be the set of k members of $G(v) \cup B(v)$ with smallest key values where key is $g(u) - h(u)$ for $u \in G(v)$ and $b(u) - h(u)$ for $u \in B(v)$. Otherwise $S''(v) = \emptyset$. Now we have following lemma which leads to a simple dynamic program for PDBEP.

Lemma 10. *For any internal vertex v of T,*

(i) $h(v) = \sum_{u \in B(v)} b(u) + \sum_{u \in H(v)} h(u) + \sum_{u \in G(v)} g(u) + \min\{c_v - 1, |H(v)| + |B(v)|\}$.
If $d(v) = c_v$ *and* $v \neq R$, *then set* $b(v) = g(v) = 0$ *otherwise*
(ii) $g(v) = \sum_{u \in B(v)} b(u) + \sum_{u \in H(v)} h(u) + \sum_{u \in G(v) \setminus S'(v)} g(u) + \sum_{u \in S'(v)} \max\{h(u), b(u)\} + c_v$.
(iii) $b(v) = \sum_{u \in H(v) \cup S''(v)} h(u) + \sum_{u \in B(v) \setminus S''(v)} b(u) + \sum_{u \in G(v) \setminus S''(v)} g(u) + \max\{c_v, |H(v)|\}$.

Observe that if $h(u)$ is equal to $b(u)$ or $g(u)$, then u is categorized as an $H(v)$ vertex and if $b(u) = g(u) > h(u)$, then u is assigned to $B(v)$ set. Hence the last term is maximum in each of the cases in the lemma.

The algorithm initializes $h(v), b(v)$, and $g(v)$ to zero for the leaf vertices and computes these values for the internal vertices bottom up. Finally it outputs the maximum of the three values of the root R. In order to compute $S'()$ and $S''()$ sets for each vertex, we need to sort the child nodes with respect to the key values. Thus at each vertex we incur $O(|Ch| \log |Ch|)$ cost, where Ch denotes the set of children of that vertex. Besides, ordering the vertices so that child occurs before the parent (topological sort) takes $O(n)$ time. Hence the time complexity is $O(n \log n)$.

Acknowledgement: *We thank the referees of the paper for detailed feedback and suggestions which improved the analysis of the weighted case and also the overall presentation of the paper.*

References

1. Frank Dehne, Michael Fellows, Henning Fernau, Elena Prieto, and Frances Rosamond. nonblocker: Parameterized algorithmics for minimum dominating set. In Jiří Wiedermann, Gerard Tel, Jaroslav Pokorný, Mária Bieliková, and Július Štuller, editors, *SOFSEM 2006: Theory and Practice of Computer Science*, volume 3831 of *Lecture Notes in Computer Science*, pages 237–245. Springer Berlin Heidelberg, 2006.
2. Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
3. J. Nieminen. Two bounds for the domination number of a graph. *Journal of the Institute of Mathematics and its Applications*, 14:183–187, 1974.
4. Peng Zhang. Partial degree bounded edge packing problem. In *Proceedings of the 6th international Frontiers in Algorithmics, and Proceedings of the 8th international conference on Algorithmic Aspects in Information and Management*, FAW-AAIM'12, pages 359–367. Springer-Verlag, 2012.