Partial Degree Bounded Edge Packing Problem for Graphs and k-Uniform Hypergraphs

Pawan Aurora Sumit Singh
Shashank K Mehta
Indian Institute of Technology, Kanpur - 208016, India
paurora@iitk.ac.in,ssumit@iitk.ac.in,skmehta@cse.iitk.ac.in

January 29, 2015

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). In this work we study the general degree constraint case (arbitrary degree constraint for each vertex) and present two combinatorial approximation algorithms with approximation factors 4 and 2. Then we give a $\log_2 n$ approximation algorithm for edge-weighted version of the problem and an efficient exact algorithm for edge-weighted trees with time complexity $O(n \log n)$. We also consider a generalization of this problem to k-uniform hypergraphs and present a constant factor approximation algorithm based on linear programming using Lagrangian relaxation.

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

1 Introduction

A (k, t)-*partial degree bounded edge packing problem* ((k, t)-PDBEP) is described as follows: Given a k-uniform hypergraph $G = (V, E)$ and a *degree-bound function* $c : V \to \mathbb{Z}_{\geq 0}$, compute a maximum cardinality set $E' \subseteq E$ such that every (hyper)edge satisfies the *degree condition* in the (hyper)graph $G' = (V, E')$. An edge e in G' is said to satisfy the degree condition if there are at least t vertices, say, $u_1, \ldots, u_t$ in e such that $d'(u_i) \leq c(u_i)$. Here $d'(x)$ denotes the degree of vertex x in graph G'.

Without loss of generality we assume that $c(x) \leq d(x)$ for all $x \in V$. For convenience we will use notation c_v in place of $c(v)$. We further assume that $t \geq 1$ since $t = 0$ is a trivial case.

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 same condition as described above.

Zhang [12] studied the $(2,1)$-PDBEP problem with constant function c. Note that $(2,1)$-instance corresponds to normal graphs. This 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 $(2,1)$-PDBEP problem with constant degree-bound function $c = 2$. The generalized problem (k,t)-PDBEP finds another interesting application in resource allocation. Given n resources and m jobs, each job requires k resources. Associated with each resource u there is an integer c_u. Suppose a set of jobs J' is to be scheduled for concurrent processing. A resource u is said to be *easily accessible* in this schedule if the number of jobs in J' requiring u is no more than c_u. The set J' can be processed concurrently if for each job in J' there are at least t easily accessible resources among the k resources required by it. The objective is to compute the largest subset of jobs that can be processed concurrently.

1.1 Related Work

The decision version of $(2,1)$-PDBEP problem when there is a constant degree bound function $c = 1$ is the parametric dual of the Dominating Set (DS) problem. It was studied by Nieminen [9]. It was also studied under the framework of parameterized complexity by Dehne, Fellows, Fernau, Prieto and Rosamond in [2].

Recently Peng Zhang [12] showed that the $(2,1)$-PDBEP problem with constant degree bound function is NP-hard even for $c = 1$. They gave approximation algorithms for $(2,1)$-PDBEP problem for the degree bound functions $c = 1$ and $c = 2$ with approximation factors 2 and $32/11$ respectively. They showed that $(2,1)$-PDBEP problem on trees with constant degree bound function can be solved exactly in $O(n^2)$ time.

1.2 Our Contribution

We propose three different approximation algorithms for the unweighted problem with arbitrary degree bound function (i.e., for arbitrary non-negative function c). Two of these algorithms are combinatorial in nature and give approximation ratios of 4 and 2 for $(2,1)$-PDBEP. The third algorithm solves the general (k,t)-PDBEP. Here we start with the natural IP formulation (IP1) which turns out to have a large

integrality gap. To overcome this we propose a Lagrangian-like relaxation where we give an approximate IP (IP2) such that any α approximation of this IP is a $\alpha(k-t+1)$ approximation of (k,t)-PDBEP problem. Lagrangian relaxation has been used extensively in the design of approximation algorithms for a variety of problems (see e.g., [1, 3, 10, 7, 6, 5] and a comprehensive survey in [8]). We then present a $t(k-t+1)(1-1/k)+1/k$ approximation iterative rounding algorithm for IP2. Putting these results together we get $t(1-1/k)(k-t+1)^2+(k-t+1)(1/k)$ approximation for (k,t)-PDBEP problem. See [4] for details on iterative rounding.

Next we consider the $(2,1)$-PDBEP problem for edge-weighted graphs with arbitrary degree bound function. In this case we present a combinatorial approximation algorithm with approximation factor of $2+\log_2 n$. Edge-weighted $(2,1)$-PDBEP problem is not addressed in the literature, to the best of our knowledge.

Finally we present an exact algorithm for the $(2,1)$-PDBEP on edge-weighted trees with arbitrary degree bound function. The time complexity of this algorithm is $O(n \log n)$. This is an improvement over the $O(n^2)$ algorithm in [12] which is applicable to only the constant degree bound function case for unweighted trees.

In this work we have considered three types of generalization of PDBEP problem: (1) we have considered general function $c()$ in place of the constant function, (2) we have also considered the problem in the context of edge weighted graphs, and (3) the problem is extended to k-uniform hypergraphs, referred as (k,t)-PDBEP. In each case we have presented either an improved or a new solution.

2 Approximation Algorithms for $(2,1)$-PDBEP

The number of edges in an optimum solution of (k,t)-PDBEP problem can be bounded as follows.

Lemma 2.1. *An optimum solution of (k,t)-PDBEP has at most $(1/t)\sum_{v\in V} c_v$ edges.*

Proof. Let $E' \subset E$ be a solution of (k,t)-PDBEP. Let $U = \{v \in V | d'_v \leq c_v\}$. Then from the degree condition we see that U is a vertex cover for the hypergraph (V,E') such that each hyperedge of this hypergraph is incident on at least t vertices of U. Hence $t \cdot |E'| \leq \sum_{u\in U} c_u \leq \sum_{v\in V} c_v$. □

2.1 Edge Addition based Algorithm

Algorithm 1 computes a maximal solution Y by iteratively adding edges, i.e., in each iteration selects a new edge and adds it to Y if it does not result into degree violation on both end-vertices. By construction Y is a valid solution of $(2,1)$-PDBEP problem.

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 $E \setminus Y$ edges incident on a_1 are all distinct from those incident on a_2.

Next, the algorithm constructs 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 is also a solution of $(2, 1)$-PDBEP because every edge in it satisfies the degree constraints. Finally the larger of Y and Z is output. Either way the output is a valid solution.

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 2.1 the approximation ratio is bounded by 4.

Data: A connected graph $G = (V, E)$ and a function $c : V \to \mathbb{Z}_{\geq 0}$ 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 from E \ Y which are incident on v and
    insert them into Z;
end
if |Y| ≥ |Z| then
    return Y;
else
    return Z;
end
```

Algorithm 1: Edge Addition Based Algorithm

Theorem 2.2. *Algorithm 1 has approximation factor* 4.

2.2 Edge Deletion based Algorithm

The second algorithm for $(2,1)$-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.

Data: A connected graph $G = (V, E)$ and a function $c : V \to \mathbb{Z}_{\geq 0}$ 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

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 2.1, $|Y| \geq OPT/2$ where OPT denotes the value of the optimum solution.

Theorem 2.3. *Algorithm 2 has approximation ratio* 2.

3 Approximation Algorithm for (k,t)-PDBEP

In this section we explore a linear programming based approach to design an approximation algorithm for the (k,t)-PDBEP problem.

3.1 Integer Program

The natural IP formulation of the problem is as follows. Here x_u takes value 1 if u satisfies degree condition. Otherwise it takes value zero.

$$
\begin{aligned}
\text{IP1:}\quad \text{maximize} \quad & \psi = \sum_{e \in E} y_e \\
\text{subject to} \quad & y_e \leq \frac{1}{t}(x_{u_1} + x_{u_2} + \ldots + x_{u_k}) \ , \forall e = \{u_1, \ldots, u_k\} \in E \\
& \sum_{e \in \delta(v)} y_e \leq c_v x_v + d_v(1 - x_v) \quad , \forall v \in V \\
& x_v \in \{0, 1\} \qquad\qquad , \forall v \in V \\
& y_e \in \{0, 1\} \qquad\qquad , \forall e \in E
\end{aligned}
$$

In the above $\delta(v)$ denotes the set of edges incident on vertex v. 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 3.1. *The integrality gap of IP1 is $\Omega(n)$ where n is the number of vertices in the hypergraph.*

Proof. Consider the following instance of the problem for normal graphs, i.e., $k = 2, t = 1$. Let G be a complete graph on n vertices $\{v_0, v_1, \ldots, v_{n-1}\}$ and the degree bound 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 v and $y_e = 1$ for $e = (v_i, v_j)$ for all i, 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 2.1, the optimum solution for the IP1 cannot be more than n. Hence the integrality gap is $\Omega(n)$. □

High integrality gap necessitates an alternative approach.

3.2 Approximate Integer Program

We propose an alternative integer program IP2 given below where ϵ is any positive real number. It 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 $\alpha(k - t + 1)/(1 - \epsilon(k - t))$ approximation of IP1. A solution of IP2 is *maximal* if perturbing the value of any y_e or any z_v either renders the solution infeasible or does not improve its objective function value.

$$\begin{aligned}
\text{IP2:}\quad \text{maximize}\quad & \phi = (k-t+1)\sum_{e\in E} y_e - (1+\epsilon)\sum_{v\in V} z_v & \\
\text{subject to}\quad & \sum_{e\in\delta(v)} y_e \le c_v + z_v & , \forall v \in V \\
& z_v \in \{0,1,2,\ldots\} & , \forall v \in V \\
& y_e \in \{0,1\} & , \forall e \in E
\end{aligned}$$

Note that any feasible solution of IP1 is also 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, in any maximal solution of IP2, $z_v = \max\{0, \sum_{e\in\delta(v)} y_e - c_v\}$. Hence in a maximal solution z values need not be specified. We will denote $\sum_v z_v$ by Z. Given any valid solution E', $\phi(E')$ will denote the objective function value where $y_e = 1$ if and only if $e \in E'$ and $z_v = \max\{0, \sum_{e\in\delta(v)} y_e - c_v\}$ for all v.

Lemma 3.2. *Every maximal solution of the integer program IP2 is also a feasible solution of (k,t)-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 (k,t)-PDBEP. Then there must exist an edge $e = \{u_1, \ldots, u_k\} \in E'$ such that more than $k-t$ vertices incident on e are in violation. Suppose that number is $k-t+q$. Let $E'' = E' \setminus \{e\}$, i.e., set y_e to zero. Then for each vertex $u \in e$ and having $z_u > 1$, we can reduce z_u by 1 and still get a valid solution. Then $\phi(E'') - \phi(E') \ge (k-t+q)(1+\epsilon) - (k-t+1) > 0$ because $q \ge 1$. This contradicts the fact that E' is a maximal solution. □

In a maximal solution E' of IP2, from the point of view of IP1, Z is the sum of excess degrees of violating vertices. Hence $Z \le (k-t)|E'|$.

Lemma 3.3. *Any α approximate solution of IP2, which is also maximal, is a $\alpha(k-t+1)/(1-\epsilon(k-t))$ approximation of (k,t)-PDBEP problem.*

Proof. Let E'_1 be an optimum solution of (k,t)-PDBEP and E'_2 be an α - approximation maximal solution of IP2. Then E'_1 is also a solution of IP2 with $y_{1e} = 1$ for $e \in E'_1$ and $z_{1v} = \max\{0, \sum_{e\in\delta(v)} y_{1e} - c_v\}$. Then $\phi(E'_1) = (k-t+1)|E'_1| - (1+\epsilon)Z_1$ and $\phi(E'_2) = (k-t+1)|E'_2| - (1+\epsilon)Z_2$. Let OPT denote the optimum value of IP2. Then $\phi(E'_1) \le OPT$ and $OPT/\alpha \le \phi(E'_2)$. So $(k-t+1)|E'_1| - (1+\epsilon)Z_1 \le \alpha((k-t+1)|E'_2| - (1+\epsilon)Z_2)$. Let $\beta_1 = Z_1/|E'_1|$ and $\beta_2 = Z_2/|E'_2|$. Then $|E'_1|/|E'_2| \le \alpha(k-t+1-(1+\epsilon)\beta_1)/(k-t+1-(1+\epsilon)\beta_2) \le \alpha(k-t+1)/(1-\epsilon(k-t))$ because β_i are between 0 and $k-t$.

□

3.3 Algorithm for IP2

We propose Algorithm 3 which approximates the IP2 problem with approximation factor $t(k-t+1)(1-1/k)+1/k$. LP2 is the linear program relaxation of IP2. Here we assume an additional set of constraints, $\{z_v = 0 | v \in C\}$, where we require a solution in which every $v \in C$ must necessarily satisfy the degree condition. 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 remove 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 after Lemma 3.4). To simplify the analysis, Algorithm 3 is presented in the recursive format.

$$
\begin{aligned}
\text{LP2:} \quad \text{maximize} \quad & \phi = (k-t+1)\sum_{e \in E} y_e - (1+\epsilon)\sum_{v \in V} z_v & \\
\text{subject to} \quad & \sum_{e \in \delta(v)} y_e \leq c_v + z_v & , \forall v \in V \setminus C \\
& \sum_{e \in \delta(v)} y_e \leq c_v & , \forall v \in C \\
& z_v \geq 0 & , \forall v \in V \\
& y_e \geq 0 & , \forall e \in E \\
& y_e \leq 1 & , \forall e \in E
\end{aligned}
$$

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 of *SolveIP2* function. For simplicity we will refer to them as the problems associated with graphs H and H_1 respectively.

Lemma 3.4. *In a corner solution of LP2 on a non-empty graph there is at least one edge e with $y_e = 0$ or $y_e \geq 1/k$.*

Proof. Assume the contrary, i.e., in an extreme point solution of LP2 all y_e are in the open interval $(0, 1/k)$. 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 $\sum_{e \in \delta(v)} y_e \leq c_v + z_v$ or $\sum_{e \in \delta(v)} y_e \leq c_v$ (depending on $v \in V \setminus C$ or $v \in C$) tight (an equality), so $n_i'' = n_i - n_i'$ vertices have the condition a strict inequality. Let the number of edges in H be m.

The total number of variables is $n_1 + n_2 + n_3 + m$. In $n_1' + n_2'$ cases $\sum_{e \in \delta(v)} y_e = c_v + z_v$ or $\sum_{e \in \delta(v)} y_e = c_v$ (depending on $v \in V \setminus C$ or $v \in C$) where $c_v \geq 1$

and each $y_e < 1/k$ so there must be at least $k+1$ edges incident on such vertices. Since the graph has no isolated vertices, every vertex has at least one incident edge. Hence $m \geq n_1' + n_2' + (n_1 + n_2 + n_3)/k$. So the number of variables is at least $n_1' + n_2' + (1 + 1/k)(n_1 + n_2 + n_3)$.

Now we find the number of tight conditions. No y_e touches zero or one. The number of z_v which are equal to zero is n_2, and the number of instances when $\sum_{e \in \delta(v)} y_e = c_v + z_v$ or $\sum_{e \in \delta(v)} y_e = c_v$ (depending on $v \in V \setminus C$ or $v \in C$) 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 $n_2 + n_1' + n_2' + n_3' \geq n_1' + n_2' + (1+1/k)(n_1 + n_2 + n_3)$. 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 optimum (corner) 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 iteration at most $|V|$ edges will remain in the residual graph. Thus the total number of iterations cannot exceed $|V| + 1$.

Lemma 3.5. *If $y_e > 0$ in any maximal solution of LP2 where $e = \{u_1, \ldots, u_k\}$, then $(c_{u_i} > 0, z_{u_i} = 0)$ for at least t vertices in e.*

Proof. Let $S = \{u \in e | z_u > 0\}$. Assume that $|S| \geq k - t + 1$. Let $\delta = \min\{y_e, min_{u \in S} \{z_u\}\}$. Subtract δ from y_e and each z_u for $u \in S$. The resulting solution is still a feasible solution of LP2 and its objective function value is greater than the optimum by at least $\epsilon \cdot \delta(k - t + 1)$. This is absurd. Hence z_u cannot be positive for more than $k - t$ vertices in any edge e having $y_e > 0$ in the solution.

Next assume that $c_u = 0$ for some $u \in e \setminus S$. Then y_e must be zero, contradicting the fact that $y_e > 0$.

Therefore at least t vertices in e must have $(c_{u_i} > 0, z_{u_i} = 0)$. □

Lemma 3.6. *Algorithm 3 returns a feasible solution of (k, t)-PDBEP.*

Proof. Let H denote the input graph in the i-th iteration of the algorithm, for some i, and I denote the computed solution for it. Using induction we will show that I is a solution of (k, t)-PDBEP on H. Actually we will establish a stronger claim: I is a solution of (k, t)-PDBEP on H and every vertex in input set C satisfies the degree condition. In the following, parameters associated with the $i+1$-st iteration will be expressed with subscript-1, for example, H_1, I_1, f_{u1} etc.

In the base case (last iteration) the graph has no edges hence the solution $I = \emptyset$ trivially satisfies the claim.

In the induction step the input graph is H and the computed solution is I in i-th iteration, for some i. From induction hypothesis, the $i+1$-step solution I_1 is a feasible (k,t)-PDBEP solution on graph H_1 and every vertex in C_1 satisfies the degree constraint with respect to I_1.

First consider the case when $y_e = 0$. In this case $f_u = f_{u1}$ for all $u \in V$, $H = H_1 \cup \{e\}$, $C = C_1$, $I = I_1$. Hence I is a solution of PDBEP on H and every vertex in C satisfies the degree condition.

Now we consider the case when $y_e \geq 1/k$. We have to show that each edge in I is valid w.r.t. I, i.e., at least t vertices incident on each edge satisfy the degree condition in I. We consider three cases.

1. Consider some $u \notin e$. Then $\delta(u) = \delta_1(u)$ and $f_u = f_{u1}$. So u satisfies the degree condition in I if and only if it satisfies the same in I_1. So every $e' \in I$ such that $e' \cap e = \emptyset$, is valid in I.

2. Let $u_1, \ldots, u_t$ be the vertices which are selected, in the algorithm, from e to be inserted in C to form C_1. Then $f_{u_i} > 0$ and $z_{u_i} = 0$ and $f_{u_i 1} = f_{u_i} - 1$ for $1 \leq i \leq t$. From induction hypothesis each u_i satisfies degree condition in I_1. Coming back to I, the degree and f-value of these vertices increase by 1. So they continue to satisfy the degree condition in I. Hence e is valid in I.

3. Now we have to consider $e' \in I$ which are different from e but $e' \cap e \neq \emptyset$. Clearly $e' \in I_1$, so it is valid in I_1 from induction hypothesis. Assume that there is a vertex $u \in e' \cap e$ which satisfies the degree condition in I_1 but does not satisfy the condition in I. This is possible only if $f_u = f_{u1}$. But this happens only if $f_u = f_{u1} = 0$. But $e' \in I_1$ and it is incident on u so u was not satisfying degree condition in I_1, a contradiction. So e' continues to remain valid in I.

We conclude that I is a solution of (k,t)-PDBEP for H. Lastly we have to show that every vertex in C satisfies the degree condition with respect to I.

From the induction hypothesis each $v \in C$ satisfies the condition in I_1 because $C \subseteq C_1$. First consider the case that $v \notin e$. In this case $f_v = f_{v1}$ and the degree of v in I is same as in I_1. So v continues to satisfy the degree condition in I. Next consider the case that $v \in e$. Since $z_v = 0$ and $\sum_{e' \in \delta(v)} y_{e'} \leq f_v$, $f_v > 0$. So $f_{v1} = f_v - 1$. Now f-value and the degree of v both increase when we go from I_1 to I. As v was satisfying the degree condition in I_1, it still satisfies it in I. □

Now we analyze the performance of the algorithm.

Lemma 3.7. *Algorithm 3 gives a $(t(k-t+1)(1-1/k)+1/k)/(1-\epsilon(k-t))$ approximation for IP2.*

Proof. Let us define $c = (t(k-t+1)(1-1/k)+1/k)/(1-\epsilon(k-t))$. Let E' denote the optimum solution of LP2 on H and I denote the solution computed by the algorithm for IP2. We can also treat I as a solution of LP2 by assigning

Data: A connected hypergraph $G = (V, E)$ and a function $c : V \to \mathbb{Z}_{\geq 0}$
Result: A solution of IP2.
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

$z_u = \max\{0, \sum_{e' \in \delta(u)} y_{e'} - f_u\}$. Similarly define E'_1 and I_1 for H_1. Our goal is to show that $c\phi(I) \geq \phi(E')$. This claim is trivially true for the base case. From induction hypothesis $c\phi(I_1) \geq \phi(E'_1)$. We consider two cases: (i) $y_e = 0$, (ii) $y_e \geq 1/k$. In the first case $\phi(I) = \phi(I_1)$ and $\phi(E') = \phi(E'_1)$ so $c\phi(I) \geq \phi(E')$ trivially holds. Next we consider the second case.

Let us begin with the solution E' of LP2 for H and construct a solution of LP2 for H_1. Let $\alpha = y_e$. For each $u \in e$ such that $f_u > 0$, the f-value is decreased by 1 as we go to the next iteration. So we need to decrease the value of $\sum_{e' \in \delta(u)} y_{e'}$ suitably. First we remove y_e from the solution. That balances α. Now if the remainder of $y_{e'}$ add up to at least $1 - \alpha$, then decrease the values of these $y_{e'}$ in any way so that total decrement is $1 - \alpha$. If their sum is less than $1 - \alpha$, then set those $y_{e'}$ to zero. Repeat this step on each $u \in e$ with positive f-value. In case where $f_u = 0$ for some $u \in e$, subtract z_u by α. The resulting solution is a valid solution of LP2 for H_1, call it E''_1.

Let n_1 be the number of e vertices with f-value zero in H. From the fact that E'_1 is the optimum solution of LP2 for H_1 and the details of the construction of E''_1 we have $\phi(E'_1) \geq \phi(E''_1) \geq \phi(E') + n_1(1+\epsilon)\alpha - (k-t+1)(\alpha + (k-n_1)(1-\alpha))$. On the other hand $\phi(I) \geq \phi(I_1) + (k-t+1) - n_1(1+\epsilon)$.

From induction hypothesis $c\phi(I_1) \geq \phi(E'_1)$. So $c\phi(I) \geq \phi(E') + n_1(1+\epsilon)\alpha - (k-t+1)(\alpha+(k-n_1)(1-\alpha)) + c((k-t+1) - n_1(1+\epsilon))$. Since the coefficient of α is positive, set it to its least value, namely, $1/k$. So $c\phi(I) \geq \phi(E') + (1/k)n_1(1+\epsilon) - (1/k)(k-t+1)(1+(k-n_1)(k-1)) + c((k-t+1) - n_1(1+\epsilon))$. In this expression the coefficient of n_1 is negative so we replace it by $k - t$, its largest possible value. So $c\phi(I) \geq \phi(E') + c(1 - \epsilon(k-t)) + (k-t)(1+\epsilon)/k - (k-t+1)(1/k)(1+t(k-1))$. This may be rewritten as $c\phi(I) \geq \phi(E') + c(1-\epsilon(k-t)) - (1-\epsilon(k-t))/k - (k-t+1)t + (k-t+1)t/k$. So $c\phi(I) \geq \phi(E') + c(1 - \epsilon(k-t)) - 1/k - (k-t+1)t + (k-t+1)t/k$. Plugging the value of c we get $c\phi(I) \geq \phi(E')$. □

Combining lemmas 3.3 and 3.7 we have the following result.

Function: SolveIP2($H = (V_H, E_H), C, f$)
if $E_H := \emptyset$ **then**
  return $\emptyset$;
end
$V_H := V_H \setminus \{v | v \text{ is isolated in } H\}$;
$(\mathbf{y}, \mathbf{z}) = \text{LPSolver}(H, C)$;
/* solve LP2 with degree-bounds $f(x)$ for all $x \in V_H$ */
if $\exists e \in E_H$ *with* $y_e = 0$ **then**
  $H_1 := (V_H, E_H \setminus \{e\})$;
  $C_1 := C$;
  $E' := SolveIP2(H_1, C_1, f)$;
else
  From Lemma 3.4 there exists an edge $e := \{u_1, \ldots, u_k\}$ with $y_e \geq 1/k$;
  From Lemma 3.5 w.l.o.g. we assume $(f_{u_i} > 0, z_{u_i} = 0)$ for $i = 1, \ldots, t$;
  for $j = 1$ *to* t **do**
    $f_{u_j} := f_{u_j} - 1$;
  end
  $C_1 := C \cup \{u_1, \ldots, u_t\}$;
  for $j = t + 1$ *to* k **do**
    $f_{u_j} := \max\{f_{u_j} - 1, 0\}$;
  end
  $H_1 := (V_H, E_H \setminus \{e\})$;
  $E' := SolveIP2(H_1, C_1, f) \cup \{e\}$;
  /* Including e in E' means y_e is rounded up to 1. In case $f_{u_i} = 0$, z_{u_i} is implicitly raised to ensure that $\sum_{e' \in \delta(u_i)} y_{e'} \leq f_{u_i} + z_{u_i}$ continues to hold. We do not explicitly increase z_{u_i} value in the code since it is not output as a part of the solution. */
end
return E';

Theorem 3.8. *Algorithm 3 approximates (k,t)-PDBEP with approximation factor* $\frac{t(1-1/k)(k-t+1)^2+(k-t+1)(1/k)}{(1-\epsilon(k-t))^2}$.

4 Approximation Algorithm for Edge-Weighted $(2,1)$-PDBEP

In this section we will present an approximation algorithm for edge weighted $(2,1)$-PDBEP with arbitrary degree bound function.

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 2.1 the optimum solution of $(2,1)$-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 up to $1+\log_2|V|$ solutions, which cover $\cup_{v\in V}H(v)$. Then the heaviest solution gives a $2+\log_2|V|$ approximation of the problem.

4.1 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. Observe that the graph has no directed cycle since all edge weights are distinct.

Step 4: Label the vertices with integers 0 to $n-1$ such that if edge (u,v) is directed from u to v, then $\mathit{Label}(u) < \mathit{Label}(v)$. 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 $\log_2 n+1$ sets, $T, A_0,\ldots,A_{k-1}$, which has maximum cumulative edge weight.

Theorem 4.1. *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 $OPT/(2 + \log_2 n)$. □

5 Exact Algorithm for Edge-Weighted Trees

In this section we give a polynomial time exact algorithm for the edge-weighted $(2, 1)$-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 $(2, 1)$-PDBEP for tree $T(v)$ in which the degree of v is at most $c_v - 1$, call them H-solutions (white). Let $h(v)$ denote the weight (sum of the weights of the edges) of the maximum-weight solution among the H-solutions. Similarly let $g(v)$ be the maximum-weight G-solution (grey) in which the degree of v is restricted to be equal to c_v. Lastly $b(v)$ will denote the maximum-weight 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 its 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 child u of v in the solution has $d'(u) \leq c_u$. If in any of these categories there are no feasible solutions, then the corresponding maximum-weight value is assumed to be zero. Hence the optimum solution of $(2, 1)$-PDBEP for T is the maximum of $h(R), g(R)$, and $b(R)$. Note that all the three values are zero for leaf nodes because the corresponding tree has no edges. In this algorithm we will show how to compute these three values, $h(x), g(x)$ and $b(x)$ for each vertex x, from bottom up. Finally we output $\max\{h(R), g(R), b(R)\}$.

Suppose we know the three values of every vertex in $T(v)$, except v. Our objective is to compute these values for v. Let $Ch(v)$ denote the set of child-nodes of v. We partition $Ch(v)$ into $H(v) = \{u \in Ch(v) | h(u) \geq \max\{g(u), b(u)\}\}$, $G(v) = \{u \in Ch(v) | g(u) > \max\{h(u), b(u)\}\}$, $B(v) = Ch(V) \setminus (G(v) \cup H(v))$. Note that in case $b(u) = g(u) > h(u)$, then u is placed in $B(v)$.

Our goal is to construct one optimum member of each of H-, G-, and B-solutions of $T(v)$. The construction of the maximum-weight solution tree, in each case, in-

volves deciding which edges from v to its children must be a part of the solution tree. Further we have to decide which solution of each child will be included in the solution of $T(v)$. It is easy to observe that if we decide not to include (v, u) in the solution, then we must pick the optimum solution of $T(u)$, i.e., which corresponds to the largest of $h(u), g(u)$, and $b(u)$. Note that the weight of the optimum solution of vertex u belonging to $H(v)$, $G(v)$, and $B(v)$ is $h(u), g(u)$, and $b(u)$ respectively. This is so because not adding the edge (v, u) to the solution leaves the degree of u unchanged, i.e., if u was not violating the degree condition, then it continues to do so. Similarly, if we decide to include the edge (v, u) and $u \in H(v) \cup B(v)$, then also we can use the optimum solution of $T(u)$.

Suppose A is the optimum solution of $T(u)$ and let B be the best solution of $T(u)$ that can be used if we decide to include the edge (v, u). Then the contribution of u to the solution of $T(v)$ being constructed is A if the edge (v, u) is not included, and it is $B + w(v, u)$ if the edge is included. We define $gain(u) = B + w(v, u) - A$, which is the gain achieved if the edge (v, u) is included.

In order to construct a maximum weight G-solution of $T(v)$, connecting v to any $u \in H(v) \cup B(v)$ we get a contribution of $w(v, u) + \max\{h(u), b(u)\}$ because in this case we can use the optimum solution of $T(u)$. If we do not connect v to such a vertex, then the contribution will be only $\max\{h(u), b(u)\}$. Hence the $gain(u)$ will be $w(v, u)$. Now if $u \in G(v)$ and if we connect it with v, then we can only use H-solution or B-solution of u because using a G-solution will result in $d'(u) = c_u + 1$ and some child node of u may not be satisfying the degree condition. If all children of u are satisfying degree condition in the optimum G-solution, then this solution is also included in B-solutions. Therefore in case $u \in G(v)$, not connecting v with u gives a contribution of $g(u)$. But connecting with v gives $w(v, u) + \max\{h(u), b(u)\}$. Hence the $gain(u)$ is $w(v, u) + \max\{h(u), b(u)\} - g(u)$. Now that we know the gain for each $u \in Ch(v)$, we sort the vertices of $Ch(v)$ in non-increasing order of their $gain()$. To construct a maximum weight G-solution of v, we select top c_v vertices of the sorted list, call it set $S'(v)$, and connect v with them.

Next consider the construction of a maximum weight B-solution of $T(v)$. We can connect v to any number of $H(v)$ vertices and use their optimum H-solutions. Connecting v to each such vertex u will give a gain of $w(v, u)$. Next if $u \in B(v) \cup G(v)$, then also we can only use its H-solution. In this case $gain(u)$ will be $w(v, u) + h(u) - \max\{g(u), b(u)\}$. Once agin we sort the vertices of $Ch(v)$ in non-increasing order of the gains associated with them. Let first k_1 vertices have positive gain and the rest have non-positive gain. The subset $S''(v)$ of $Ch(v)$ to which v should be connected is computed as follows. If $c_v \leq k_1$, then $S''(v)$ includes all the top k_1 vertices. Otherwise it includes top c_v vertices.

In the construction of a maximum weight H-solution of $T(v)$, observe that an edge between v and any $u \in H(v) \cup B(v)$ can be included and the optimum solution

of $T(u)$ can be used. In this case $gain(u) = w(v,u)$. But for $u \in G(v)$ if the edge (v,u) is included in the solution, then the optimum solution of $T(u)$ (which is a G-solution) cannot be included in the solution being formed. Hence the net gain on including (v,u) in this case is $w(v,u) + \max\{h(u), b(u)\} - g(u)$. Once again we sort the elements of $Ch(v)$ in non-increasing order of their gain. Suppose first k_1 vertices have positive gain and the rest have non-positive gain. First $\min\{k_1, c_v - 1\}$ vertices, denoted by set $S'''(v)$ are connected with v.

Lemma 5.1. *For any internal vertex v of T,*

(i) $h(v) = \sum_{u \in B(v)} b(u) + \sum_{u \in H(v)} h(u) + \sum_{u \in G(v)} g(u) + \sum_{u \in S'''(v)} gain(u)$, *where* $gain(u) = w(v,u) + \max\{h(u), b(u)\} - g(u)$ *for* $u \in G(v)$ *and* $gain(u) = w(v,u)$ *for* $u \in B(v) \cup H(v)$.
If $d(v) = c_v$ *and* $v \neq R$, *then set* $b(v) = g(v) = 0$ *otherwise*

(ii) $b(v) = \sum_{u \in B(v)} b(u) + \sum_{u \in H(v)} h(u) + \sum_{u \in G(v)} g(u) + \sum_{u \in S''(v)} gain(u)$, *where* $gain(u) = w(v,u) + h(u) - \max\{g(u), b(u)\}$ *for* $u \in G(v) \cup B(v)$ *and* $gain(u) = w(v,u)$ *for* $u \in H(v)$.

(iii) $g(v) = \sum_{u \in B(v)} b(u) + \sum_{u \in H(v)} h(u) + \sum_{u \in G(v)} g(u) + \sum_{u \in S'(v)} gain(u)$, *where* $gain(u) = w(v,u) + \max\{h(u), b(u)\} - g(u)$ *for* $u \in G(v)$ *and* $gain(u) = w(v,u)$ *for* $u \in B(v) \cup H(v)$.

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 $S'(), S''()$ and $S'''()$ sets for each vertex, we need to sort the child nodes with respect to the gain 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 $O(n \log n)$.

6 Conclusion

In this work we gave constant factor approximation algorithms for the $(2,1)$-PDBEP problem. These algorithms are both simple and efficient taking time linear in the size of the input. However their analysis does not seem to extend easily to the general (k,t)-instance of the problem. Next we gave a constant factor (taking k, t as constants) approximation algorithm for the (k,t)-PDBEP employing LP based approach which uses the techniques of Lagrangian relaxation and iterative rounding. This is an expensive algorithm that needs to solve a linear program several times. However, it shows the power of linear programming based approach in combinatorial optimization. Subsequently we presented a $\log_2 n$ approximation algorithm

for the $(2,1)$-PDBEP where edges are weighted. Finally an exact algorithm for edge-weighted trees is presented with time complexity $O(n \log n)$.

Weighted (k,t)-PDBEP is an open problem. The objective function of LP1 can be easily modified to handle the weighted case, but due to the large integrality gap it remains useless. However, there are cutting-plane methods like Chvatal-Gomory cuts [11] that have been known to improve the integrality gaps for some problems. It would be worthwhile to see if these methods can help reduce the integrality gap of our LP.

As far as we know there is no known inapproximability result for the PDBEP problem. So that presents another avenue for further research.

Acknowledgement: *We thank the referees of the paper for detailed feedback and suggestions which improved the overall presentation of the paper. We also thank an anonymous referee for pointing out an error in lemma 3.4.*

References

[1] André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. In *IPCO*, pages 273–287, 2008.

[2] 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.

[3] Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *FOCS*, pages 302–309, 1996.

[4] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[5] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[6] Jochen Könemann, Ojas Parekh, and Danny Segev. A unified approach to approximating partial covering problems. *Algorithmica*, 59(4):489–509, 2011.

[7] Jochen Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM J. Comput.*, 31(6):1783–1793, 2002.

[8] J. Mestre. *Primal-Dual Algorithms for Combinatorial Optimization Problems.* PhD thesis, University of Maryland, 2007.

[9] J. Nieminen. Two bounds for the domination number of a graph. *Journal of the Institute of Mathematics and its Applications*, 14:183–187, 1974.

[10] R. Ravi and Michel X. Goemans. The constrained minimum spanning tree problem (extended abstract). In *SWAT*, pages 66–75, 1996.

[11] Mohit Singh and Kunal Talwar. Improving integrality gaps via chvátal-gomory rounding. In *APPROX-RANDOM*, pages 366–379, 2010.

[12] 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.