Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 539-550
www.stacs-conf.org

LAGRANGIAN RELAXATION AND PARTIAL COVER (EXTENDED ABSTRACT)

JULIÁN MESTRE [1]

[1] Max-Planck-Institute für Informatik, Saarbrücken, Germany.
E-mail address: jmestre@mpi-inf.mpg.de

Abstract. Lagrangian relaxation has been used extensively in the design of approximation algorithms. This paper studies its strengths and limitations when applied to Partial Cover.

We show that for Partial Cover in general no algorithm that uses Lagrangian relaxation and a Lagrangian Multiplier Preserving (LMP) α-approximation as a black box can yield an approximation factor better than $\frac{4}{3}\alpha$. This matches the upper bound given by Könemann *et al.* (*ESA 2006*, pages 468–479).

Faced with this limitation we study a specific, yet broad class of covering problems: Partial Totally Balanced Cover. By carefully analyzing the inner workings of the LMP algorithm we are able to give an almost tight characterization of the integrality gap of the standard linear relaxation of the problem. As a consequence we obtain improved approximations for the Partial version of Multicut and Path Hitting on Trees, Rectangle Stabbing, and Set Cover with ρ-Blocks.

1. Introduction

Lagrangian relaxation has been used extensively in the design of approximation algorithms for a variety of problems such as k-MST [12, 7, 11], k-median [21, 5], MST with degree constraints [27] and budgeted MST [31].

In this paper we study the strengths and limitations of Lagrangian relaxation applied to the Partial Cover problem. Let $\mathcal{S}$ be collection of subsets of a universal set U with cost $c : \mathcal{S} \to R_+$ and profit $p : U \to R_+$, and let P be a target coverage parameter. A set $\mathcal{C} \subseteq \mathcal{S}$ is a *partial cover* if the overall profit of elements covered by $\mathcal{C}$ is at least P. The objective is to find a minimum cost partial cover.

The high level idea behind Lagrangian relaxation is as follows. In an IP formulation for Partial Cover, the constraint enforcing that at least P profit is covered is *relaxed*: The constraint is multiplied by a parameter λ and lifted to the objective function. This relaxed IP corresponds, up to a constant factor, to the prize-collecting version of the underlying covering problem in which there is no requirement on how much profit to cover but a penalty

1998 ACM Subject Classification: G.2.1.
Key words and phrases: Lagrangian Relaxation, Partial Cover, Primal-Dual Algorithms.
Supported by NSF Award CCF 0430650, a University of Maryland Dean's Dissertation Fellowship, and partly by an Alexander von Humboldt Fellowship.

of $\lambda p(i)$ must be paid if we leave element $i \in U$ uncovered. An approximation algorithm for the prize-collecting version having the Lagrangian Multiplier Preserving (LMP) property[1] is used to obtain values λ_1 and λ_2 that are close together for which the algorithm produces solutions $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. These solutions are such that $\mathcal{C}_1$ is inexpensive but unfeasible (covering less than P profit), and $\mathcal{C}_2$ is feasible (covering at least P profit) but potentially very expensive. Finally, these two solutions are combined to obtain a cover that is both inexpensive and feasible.

Broadly speaking there are two ways to combine $\mathcal{C}_1$ and $\mathcal{C}_2$. One option is to treat the approximation algorithm for the prize-collecting version as a black box, only making use of the LMP property in the analysis. Another option is to focus on a particular LMP algorithm and exploit additional structure that it may offer. Not surprisingly, the latter approach has yielded better approximation guarantees. For example, for k-median compare the 6-approximation of Jain and Vazirani [21] to the 4-approximation of Charikar and Guha [5]; for k-MST compare the 5-factor to the 3-factor approximation due to Garg [12].

The results in this paper support the common belief regarding the inherent weakness of the black-box approach. First, we show a lower bound on the approximation factor achievable for Partial Cover in general using Lagrangian relaxation and the black-box approach that matches the recent upper bound of Könemann et al. [26]. To overcome this obstacle, we concentrate on Kolen's algorithm for Prize-Collecting Totally Balanced Cover [25]. By carefully analyzing the algorithm's inner workings we identify structural similarities between $\mathcal{C}_1$ and $\mathcal{C}_2$, which we later exploit when combining the two solutions. As a result we derive an almost tight characterization of the integrality gap of the standard linear relaxation for Partial Totally Balanced Cover. This in turn implies improved approximation algorithms for a number of related problems.

1.1. Related Work

Much work has been done on covering problems because of both their simple and elegant formulation, and their pervasiveness in different application areas. In its most general form the problem, also known as Set Cover, cannot be approximated within $(1-\epsilon)\ln|U|$ unless $NP \subseteq \mathrm{DTIME}(|U|^{\log\log|U|})$ [9]. Due to this hardness, easier, special cases have been studied.

A general class of covering problems that can be solved efficiently are those whose element-set incidence matrix is balanced. A $0,1$ matrix is *balanced* if it does not contain a square submatrix of odd order with row and column sums equal to 2. These matrices were introduced by Berge [4] who showed that if A is balanced then the polyhedron $\{x \geq 0 : Ax \geq 1\}$ is integral. A $0,1$ matrix is *totally balanced* if it does not contain a square submatrix with row and column sums equal to 2 and no identical columns. Kolen [25] gave a simple primal-dual algorithm that solves optimally the covering problem defined by a totally balanced matrix. A $0,\pm 1$ matrix is *totally unimodular* if every square submatrix has determinant 0 or ± 1. Although totally balanced and totally unimodular matrices are subclasses of balanced matrices, the two classes are neither disjoint nor one is included in the other.

Beyond this point, even minor generalizations can make the covering problem hard. For example, consider the *vertex cover* problem: Given a graph $G = (V, E)$ we are to choose a minimum size subset of vertices such that every edge is incident on at least one of the chosen vertices. If G is bipartite, the element-set incidence matrix for the problem

[1]The definition of the LMP property is outlined in Section 2.

is totally unimodular; however, if G is a general graph the problem becomes NP-hard [24]. Numerous approximation algorithms have been developed for vertex cover [19]. The best known approximation factor for general graphs is $2 - o(1)$ [3, 16, 23]; yet, after 25 years of study, the best constant factor approximation for vertex cover remains 2 [8, 2, 18]. This lack of progress has led researchers to seek generalizations of vertex cover that can still be approximated within twice of optimum. One such generalization is the *multicut* problem on trees: Given a tree T and a collection of pairs of vertices, a cover is formed by a set of edges whose removal separates all pairs. The problem was first studied by Garg et al. [13] who gave an elegant primal-dual 2-approximation.

A notable shortcoming of the standard set cover formulation is that certain hard-to-cover elements, also known as *outliers* [6], can render the optimal solution very expensive. Motivated by the presence of outliers, the unit-profit partial version calls for a collection of sets covering not all but a specified number k of elements. Partial Multicut, a.k.a. k-Multicut, was recently studied independently by Levin and Segev [28] and by Golovin et al. [15], who gave a $\frac{8}{3}+\epsilon$ approximation algorithm. This scheme was generalized by Könemann et al. [26] who showed how to design a $\frac{4}{3}\alpha+\epsilon$ approximation for any covering problem using Lagrangian relaxation and an α-LMP approximation as a black box. (Their algorithm runs in time polynomial on $|U|, |\mathcal{S}|^{\frac{1}{\epsilon}}$ and the running time of the α-LMP approximation.)

1.2. Our Results and Outline of the Paper

Section 3 shows that for Partial Cover in general no algorithm that uses Lagrangian relaxation and an α-LMP approximation as a black box can yield an approximation factor better than $\frac{4}{3}\alpha$. In Section 4 we give an almost tight characterization of the integrality gap of the standard LP for Partial Totally Balanced Cover, settling a question posed by Golovin et al. [15]. Our approach is based on Lagrangian relaxation and Kolen's algorithm. We prove that $\mathrm{IP} \leq \left(1 + \frac{1}{3^{k-1}}\right) \mathrm{LP} + k\, c_{\max}$ for any $k \geq 1$, where IP and LP are the costs of the optimal integral and fractional solutions respectively and $c_{\max}$ is the cost of the most expensive set in the instance. The trade-off between additive and multiplicative error is not an artifact of our analysis or a shortcoming of our approach. On the contrary, this is precisely how the integrality gap behaves. More specifically, we show a family of instances where $\mathrm{IP} > \left(1 + \frac{1}{3^{k-1}}\right) \mathrm{LP} + \frac{k}{2} c_{\max}$. In other words, there is an unbounded additive gap in terms of $c_{\max}$ but as it grows the multiplicative gap narrows exponentially fast.

Finally, we show how the above result can be applied, borrowing ideas from [14, 17, 15], to get a $\rho+\epsilon$ approximation or a quasi-polynomial time ρ-approximation for covering problems that can be expressed with a suitable combination of ρ totally-balanced matrices. This translates into improved approximations for a number of problems: a $2+\epsilon$ approximation for the Partial Multicut on Trees [28, 15], a $4+\epsilon$ approximation for Partial Path Hitting on Trees [30], a 2-approximation for Partial Rectangle Stabbing [14], and a ρ approximation for Partial Set-Cover with ρ-blocks [17]. In addition, the ϵ can be removed from the first two approximation guarantees if we allow quasi-polynomial time. It is worth noting that prior to this work, the best approximation ratio for all these problems could be achieved with the framework of Könemann et al. [26]. In each case our results improve the approximation ratio by a $\frac{4}{3}$ multiplicative factor. Due to lack of space these results only appear in the full version[2] of the paper.

[2]Full version available at `http://arxiv.org/abs/0712.3936`

2. Lagrangian relaxation

Let $\mathcal{S} = \{1, \ldots, m\}$ be a collection of subsets of a universal set $U = \{1, \ldots, n\}$. Each set has a cost specified by $c \in R_+^m$, and each element has a profit specified by $p \in R_+^n$. Given a target coverage P, the objective of the Partial Cover problem is to find a minimum cost solution $\mathcal{C} \subseteq \mathcal{S}$ such that $p(\mathcal{C}) \geq P$, where the notation $p(\mathcal{C})$ denotes the overall profit of elements covered by $\mathcal{C}$. The problem is captured by the IP below. Matrix $A = \{a_{ij}\} \in \{0,1\}^{n \times m}$ is an element-set incidence matrix, that is, $a_{ij} = 1$ if and only if element $i \in U$ belongs to set $j \in \mathcal{S}$; variable x_j indicates whether set j is chosen in the solution $\mathcal{C}$; variable r_i indicates whether element i is left uncovered.

Lagrangian relaxation is used to get rid of the constraint bounding the profit of uncovered elements to be at most $p(U) - P$. The constraint is multiplied by the parameter λ, called Lagrange Multiplier, and is lifted to the objective function. The resulting IP corresponds, up to the constant $\lambda\,(p(U) - P)$ factor in the objective function, to the prize-collecting version of the covering problem, where the penalty for leaving element i uncovered is λp_i.

$$\begin{aligned} &\min\ c \cdot x \\ &Ax + Ir \geq 1 \\ &\quad p \cdot r \leq p(U) - P \\ &\quad r_i, x_j \in \{0,1\} \end{aligned} \quad \xrightarrow[\text{Relaxation}]{\text{Lagrangian}} \quad \begin{aligned} &\min\ c \cdot x + \lambda p \cdot r - \lambda\,(p(U) - P) \\ &Ax + Ir \geq 1 \\ &\quad r_i, x_j \in \{0,1\} \end{aligned}$$

Let OPT be the cost of an optimal partial cover and OPT-PC(λ) be the cost of an optimal prize-collecting cover for a given λ. Let $\mathcal{A}$ be an α-approximation for the prize-collecting variant of the problem. Algorithm $\mathcal{A}$ is said to have the Lagrangian Multiplier Preserving (LMP) property if it produces a solution $\mathcal{C}$ such that

$$c(\mathcal{C}) + \alpha\,\lambda\big(p(U) - p(\mathcal{C})\big) \leq \alpha\,\text{OPT-PC}(\lambda). \tag{2.1}$$

Note that OPT-PC(λ) $\leq$ OPT $+ \lambda\,(p(U) - P)$. Thus,

$$c(\mathcal{C}) \leq \alpha\Big(\text{OPT} + \lambda\,\big(p(\mathcal{C}) - P\big)\Big). \tag{2.2}$$

Therefore, if we could find a value of λ such that $\mathcal{C}$ covers exactly P profit then $\mathcal{C}$ is α-approximate. However, if $p(\mathcal{C}) < P$, the solution is not feasible, and if $p(\mathcal{C}) > P$, equation (2.2) does not offer any guarantee on the cost of $\mathcal{C}$. Unfortunately, there are cases where no value of λ produces a solution covering exactly P profit. Thus, the idea is to use binary search to find two values λ_1 and λ_2 that are close together and are such that $\mathcal{A}(\lambda_1)$ covers less, and $\mathcal{A}(\lambda_2)$ covers more than P profit. The two solutions are then combined in some fashion to produce a feasible cover.

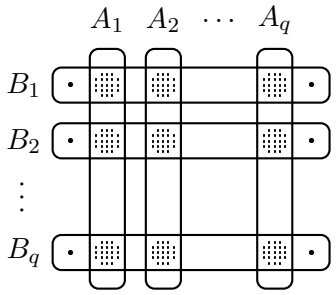
3. Limitations of the black-box approach

A common way to combine the two solutions returned by the α-LMP is to treat the algorithm as a black box, solely relaying on the LMP property (2.1) in the analysis. More formally, an algorithm for Partial Cover that uses Lagrangian relaxation and an α-LMP approximation $\mathcal{A}$ as a black box is as follows. First, we are allowed to run $\mathcal{A}$ with as many different values of λ as desired; then, the solutions thus found are combined to produce a

feasible partial cover. No computational restriction is placed on the second step, except that only sets returned by $\mathcal{A}$ may be used.

Theorem 3.1. *In general, the Partial Cover problem cannot be approximated better than $\frac{4}{3}\alpha$ using Lagrangian relaxation and an α-LMP algorithm $\mathcal{A}$ as a black box.*

Let $A_1, \ldots A_q$ and $B_1, \ldots B_q$ be sets as depicted on the right. For each i and j the intersection $A_i \cap B_j$ consists of a cluster of q elements. There are q^2 clusters. Set A_i is made up of q clusters; set B_i is made up of q clusters and two additional elements (the leftmost and rightmost elements in the picture.) Thus $|A_i| = q^2$ and $|B_i| = q^2 + 2$. In addition, there are sets $O_1, \ldots, O_q$, which are not shown in the picture. Set O_i contains one element from each cluster and the leftmost element of B_i. Thus $|O_i| = q^2 + 1$. The cost of O_i is $\frac{1}{q}$, the cost of A_i is $\frac{2\alpha}{3q}$, and the cost of B_i is $\frac{4\alpha}{3q}$. Every element has unit profit and the target coverage is $P = q^3 + q$. It is not hard to see that $O_1, \ldots, O_q$ is an optimal partial cover with a cost of 1.



The α-LMP approximation algorithm we use has the unfortunate property that it never returns sets from the optimal solution.

Lemma 3.2. *There exists an α-LMP approximation $\mathcal{A}$ that for the above instance and any value of λ outputs either $\emptyset$ or $A_1, \ldots, A_q$ or $B_1, \ldots, B_q$.*

The proof that such an algorithm exists is given in the full version of the paper. Hence, if we use $\mathcal{A}$ as a black box we must build a partial cover with the sets $A_1, \ldots, A_q$ and $B_1, \ldots, B_q$. Note that in order to cover $q^2 + q$ elements either all A-sets, or all B-sets must be used. In the first case $\frac{q}{2}$ additional B-sets are needed to attain feasibility, and the solution has cost $\frac{4}{3}\alpha$; in the second case the solution is feasible but again has cost $\frac{4}{3}\alpha$. Theorem 3.1 follows.

One assumption usually made in the literature [1, 10, 26] is that $c_{\max} = \max_j c_j \leq \epsilon\,\mathrm{OPT}$, for some constant $\epsilon > 0$, or more generally an additive error in terms of $c_{\max}$ is allowed. This does not help in our construction as $c_{\max}$ can be made arbitrarily small by increasing q.

Admittedly, our lower bound example belongs to a specific class of covering problem (every element belongs to at most three sets) and although the example can be embedded into a partial totally unimodular covering problem (see full version), it is not clear how to embed the example into other classes. Nevertheless, the $\frac{4}{3}\alpha$ upper bound of Koneman et el. [26] makes no assumption about the underlying problem, only using the LMP property (2.1) in the analysis. It was entirely conceivable that the $\frac{4}{3}\alpha$ factor could be improved using a different merging strategy—Theorem 3.1 precludes this possibility.

4. Partial Totally Balanced Cover

In order to overcome the lower bound of Theorem 3.1, one must concentrate on a specific class of covering problems or make additional assumptions about the α-LMP algorithm. In this section we focus on covering problems whose IP matrix A is totally balanced. More specifically, we study the integrality gap of the standard linear relaxation for Partial Totally Balanced Cover (P-TBC) shown below.

Theorem 4.1. *Let* IP *and* LP *be the cost of the optimal integral and fractional solutions of an instance of P-TBC. Then* $\text{IP} \leq \left(1 + \frac{1}{3^{k-1}}\right) \text{LP} + k\, c_{\max}$ *for any* $k \in Z_+$. *Furthermore, for any large enough* $k \in Z_+$ *the exists an instance where* $\text{IP} > \left(1 + \frac{1}{3^{k-1}}\right) \text{LP} + \frac{k}{2}\, c_{\max}$.

$$\begin{array}{rl} \min\ c \cdot x & \\ Ax + Ir \geq 1 & \\ p \cdot r \leq p(U) - P & \xrightarrow[\text{LP Duality}]{} \\ r_i, x_e \geq 0 & \end{array} \qquad \begin{array}{r} \max\ 1 \cdot y - (p(U) - P)\,\lambda \\ A^T y \leq c \\ y \leq \lambda p \\ y_i, \lambda \geq 0 \end{array}$$

The rest of this section is devoted to proving the upper bound in Theorem 4.1, the lower bound is left for the full version of the paper. Our approach is based on Lagrangian relaxation and Kolen's algorithm for Prize-Collecting Totally Balanced Cover (PC-TBC). The latter exploits the fact that a totally balanced matrix can be put into greedy standard form by permuting the order of its rows and columns; in fact, the converse is also true [20]. A matrix is in standard greedy form if it does not contain as an induced submatrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.1}$$

There are polynomial time algorithms that can transform a totally balanced matrix into greedy standard form [32] by shuffling the rows and columns of A. Since this transformation does not affect the underlying covering problem, we assume that A is given in standard greedy form.

4.1. Kolen's algorithm for Prize-Collecting Totally Balanced Cover

For the sake of completeness we describe Kolen's primal-dual algorithm for PC-TBC. The algorithm finds a dual solution y and a primal solution C, which is then pruned in a reverse-delete step to obtain the final solution $\widehat{C}$. The linear and dual relaxations for PC-TBC appear below.

$$\begin{array}{rl} \min\ c \cdot x + \lambda p \cdot r & \\ Ax + Ir \geq 1 & \\ r_i, x_e \geq 0 & \xrightarrow[\text{LP Duality}]{} \end{array} \qquad \begin{array}{r} \max\ 1 \cdot y \\ A^T y \leq c \\ y \leq \lambda p \\ y_i \geq 0 \end{array}$$

The residual cost of the set j w.r.t. y is defined as $c'_j = c_j - \sum_{i:a_{ij}=1} y_i$. The algorithm starts from the trivial dual solution $y = 0$, and processes the elements in increasing column order of A^T. Let i the index of the current element. Its corresponding dual variable, y_i, is increased until either the residual cost of some set j containing i equals 0 (we say set j becomes tight), or y_i equals λp_i (Lines 3-5).

Let $C = \{j \,|\, c'_j = 0\}$ be the set of tight sets after the dual update is completed. As it stands the cover C may be too expensive to be accounted for using the lower bound provided by $1 \cdot y$ because a single element may belong to multiple sets in C. The key insight is that some of the sets in C are redundant and can be pruned.

```
KOLEN((A, c, p, λ))                          ()
 1  // Dual update                           7  // Reverse delete
 2  y ← 0, C ← ∅, Ĉ ← ∅                     8  while C ≠ ∅
 3  for i ← 1 to n                          9  do j ← largest set index in C
 4  do δ ← min{c'_j | a_ij = 1}            10      Ĉ ← Ĉ + j
 5       y_i ← min{λp_i, δ}                11      C ← C \ { j' | j dominates j' or j = j' }
 6  C ← {j | c'_j = 0}                     12  return (Ĉ, y)
```

Definition 4.2. Given sets j_1, j_2 we say that j_1 *dominates* j_2 in y if $j_1 > j_2$ and there exists an item i such that $y_i > 0$ and i belongs to j_1 and j_2, that is, $a_{ij_1} = a_{ij_2} = 1$.

The reverse-delete step iteratively identifies the largest index j in C, adds j to $\widehat{C}$, and removes j and all the sets it dominates. This is repeated until no set is left in C (Lines 8–11).

Notice that all sets $j \in C$ are tight, thus we can pay for set j by charging the dual variables of items that belong to j. Because of the reverse-delete step if $y_i > 0$ then i belongs to at most one set in $\widehat{C}$; thus in paying for $\widehat{C}$ we charge covered items at most once. Using the fact A is in standard greedy form, it can be shown [25] that if i was left uncovered then we can afford its penalty, i.e., $y_i = \lambda p_i$. The solution $\widehat{C}$ is optimal for PC-TBC since

$$\sum_{j \in \widehat{C}} c_j \;+\; \sum_{\substack{i \in U \text{ s.t.} \\ \nexists\, j \in \widehat{C} : a_{ij} = 1}} \lambda p_i \;=\; \sum_{\substack{i \in U \text{ s.t.} \\ \exists\, j \in \widehat{C} : a_{ij} = 1}} y_i + \sum_{\substack{i \in U \text{ s.t.} \\ \nexists\, j \in \widehat{C} : a_{ij} = 1}} y_i \;=\; \sum_{i \in U} y_i. \tag{4.2}$$

If we could find a value of λ such that KOLEN(A, c, p, λ) returns a solution $(\widehat{C}, y)$ covering *exactly* P profit, we are done since from (4.2) it follows that

$$\sum_{j \in \widehat{C}} c_j = \sum_{i \in U} y_i - \lambda\,(p(U) - P). \tag{4.3}$$

Notice that (y, λ) is a feasible for the dual relaxation of P-TBC and its cost is precisely the right hand side of (4.3). Therefore for this instance IP=DL=LP and Theorem 4.1 follows.

Unfortunately, there are cases where no such value of λ exists. Nonetheless, we can always find a *threshold value* λ such that for any infinitesimally small $\delta > 0$, $\lambda^- = \lambda - \delta$ and $\lambda^+ = \lambda + \delta$ produce solutions covering less and more than P profit respectively. A threshold value can be found using Megiddo's parametric search [29] by making $O(n \log m)$ calls to the procedure KOLEN.

Let y (y^-) be the dual solution and C (C^-) the set of tight sets when KOLEN is run on λ (λ^-). Without loss of generality assume $\widehat{C}$ covers more than P profit. (The case where $\widehat{C}$ covers less than P profit is symmetrical: we work with y^+ and C^+ instead of y^- and C^-.)

Our plan to prove Theorem 4.1 is to devise an algorithm to merge $\widehat{C}$ and $\widehat{C}^-$ in order to obtain a cheap solution covering at least P profit.

4.2. Merging two solutions

Before describing the algorithm we need to establish some important properties regarding these two solutions and their corresponding dual solutions.

For any i, the value of y_i^- is a linear function of δ for all i. This follows from the fact that δ is infinitesimally small. Furthermore, the constant term in this linear function is y_i.

Lemma 4.3. *For each $i \in U$ there exists $a \in Z$, independent of δ, such that $y_i^- = y_i + a\delta$.*

Proof. By induction on the number of iteration of the dual update step of KOLEN, using the fact that the same property holds for the residual cost of the sets. ∎

A useful corollary of Lemma 4.3 is that $C^- \subseteq C$, since if the residual cost of a set is non-zero in y it must necessarily be non-zero in y^-. The other way around may not hold.

At the heart of our approach is the notion of a merger graph $G = (V, E)$. The vertex set of G is made up of sets from the two solutions, i.e., $V = \widehat{C} \oplus \widehat{C}^-$. The edges of G are directed and given by

$$E = \left\{ (j_1, j_2) \;\middle|\; \begin{array}{l} j_1 \in \widehat{C}^- \setminus \widehat{C},\ j_2 \in \widehat{C} \setminus \widehat{C}^- \text{ s.t. } j_1 \text{ dominates } j_2 \text{ in } y^-, \text{ or} \\ j_1 \in \widehat{C} \setminus \widehat{C}^-,\ j_2 \in \widehat{C}^- \setminus \widehat{C} \text{ s.t. } j_1 \text{ dominates } j_2 \text{ in } y \end{array} \right\} \tag{4.4}$$

This graph has a lot of structure that can be exploited when merging the solutions.

Lemma 4.4. *The merger graph $G = (V, E)$ of $\widehat{C}^-$ and $\widehat{C}$ is a forest of out-branchings.*

Proof. First note that G is acyclic, since if $(j_1, j_2) \in E$ then necessarily $j_1 > j_2$. Thus, it is enough to show that the in-degree of every $j \in V$ is at most one. Suppose otherwise, that is, there exist $j_1, j_2 \in V$ such that $(j_1, j), (j_2, j) \in E$. Assume that $j_1 < j_2$ and $j \in \widehat{C}$ (the remaining cases are symmetrical).

By definition (4.4), we know that $j_1\,(j_2) \in \widehat{C}^-$ and that there exists i_1 (i_2) that belongs to j and j_1 (j_2) such that $y^-_{i_1} > 0$ ($y^-_{i_2} > 0$). Since A^T is in standard greedy form we can infer that i_2 belongs to j_1 if $i_1 < i_2$, or i_1 belongs to j_2 if $i_1 > i_2$: The diagram on the right shows how, using the fact that A^T does not contain (4.1) as an induced submatrix, we can infer that the boxed entries must be 1. In either case we get that j_2 dominates j_1 in y^-, which contradicts the fact that both belong to $\widehat{C}^-$. ∎

	i_1	i_2	i_2	i_1
j	1	1	1	1
j_1	1	[1]		1
j_2		1	1	[1]

```
MERGE((Ĉ⁻, Ĉ))
1  let G be the merger graph for Ĉ⁻ and Ĉ
2  D ← Ĉ⁻
3  for each root r in G
4  do if p(D ⊕ T_r) ≤ P
5        then then D ← D ⊕ T_r
6             else return INCREASE(r, D)
```

The procedure MERGE starts from the unfeasible solution $D = \widehat{C}^-$ and guided by the merger graph G, it modifies D step by step until feasibility is attained. The operation used to update D is to take the symmetric difference of D and a subtree of G rooted at a vertex $r \in V$, which we denote by T_r. For each root r of an out-branchings of G we set $D \leftarrow D \oplus T_r$, until $p(D \oplus T_r) > P$. At this point we return the solution produced by INCREASE(r, D).

Notice that after setting $D \leftarrow D \oplus T_r$ in Line 5, the solution D "looks like" $\widehat{C}$ within T_r. Indeed, if all roots are processed then $D = \widehat{C}$. Therefore, at some point we are bound

to have $p(D \oplus T_r) > P$ and to make the call INCREASE(r, D) in Line 6. Before describing INCREASE we need to define a few terms. Let the *absolute benefit* of set j, which we denote by b_j, be the profit of elements uniquely covered by set j, that is,

$$b_j = p\left(\left\{\ i \in U\ \mid\ \forall\, j' \in \widehat{C} \cup \widehat{C}^{-} : a_{ij'} = 1 \text{ iff } j' = j\ \right\}\right). \tag{4.5}$$

Let $D \subseteq \widehat{C} \cup \widehat{C}^{-}$. Note that if $j \in D$, the removal of j decreases the profit covered by D by at least b_j; on the other hand, if $j \notin D$, its addition increases the profit covered by at least b_j. This notion of benefit can be extended to subtrees,

$$\Delta(T_j, D) = \sum_{j' \in T_j \setminus D} b_{j'} - \sum_{j' \in T_j \cap D} b_{j'}. \tag{4.6}$$

We call this quantity the *relative benefit* of T_j with respect to D. It shows how the profit of uniquely covered elements changes when we take $D \oplus T_j$. Note that $\Delta(T_j, D)$ can positive or negative.

Everything is in place to explain INCREASE(j, D). The algorithm assumes the input solution is unfeasible but can be made feasible by adding some sets in T_j; more precisely, we assume $p(D) \leq P$ and $P < p(D) + \Delta(T_j, D)$. If adding j to D makes the solution feasible then return $D + j$ (Lines 2-3). If there exists a child c of j that can be used to propagate the call down the tree then do that (Lines 4-5). Otherwise, *split* the subtree T_j: Add j to D and process the children of c, setting $D \leftarrow D \oplus T_c$ until D becomes feasible (Lines 6-9). At this point $p(D) > P$ and $p(D \oplus T_c) \leq P$. If $P - p(D \oplus T_c) < p(D) - P$ then call INCREASE$(c, D \oplus T_c)$ else call DECREASE(c, D) and let D' be the cover returned by the recursive call (Lines 10-12). Finally, return the cover with minimum cost between D and D'.

```
INCREASE((j, D))
 1  // assume p(D) ≤ P < p(D) + Δ(T_j, D)
 2  if p(D + j) ≥ P
 3     then return D + j
 4  if ∃ child c of j : p(D) + Δ(T_c, D) > P
 5     then return INCREASE(c, D)
 6  D ← D + j
 7  while p(D) ≤ P
 8  do c ← child of j maximizing Δ(T_c, D)
 9       D ← D ⊕ T_c
10  if P − p(D ⊕ T_c) < p(D) − P
11     then D' ← INCREASE(c, D ⊕ T_c)
12     else  D' ← DECREASE(c, D)
13  return min cost {D, D'}
```

```
DECREASE((j, D))
 1  // assume p(D) ≥ P > p(D) + Δ(T_j, D)
 2  if p((D ⊕ T_j) + j) ≥ P
 3     then return (D ⊕ T_j) + j
 4  if ∃ child c of j : p(D) + Δ(T_c, D) < P
 5     then return DECREASE(c, D)
 6  D ← D + j
 7  while p(D) ≥ P
 8  do c ← child of j minimizing Δ(T_c, D)
 9       D ← D ⊕ T_c
10  if p(D ⊕ T_c) − P < P − p(D)
11     then D' ← INCREASE(c, D)
12     else  D' ← DECREASE(c, D ⊕ T_c)
13  return min cost {D ⊕ T_c, D'}
```

The twin procedure DECREASE(j, D) is essentially symmetrical: Initially the input is feasible but can be made unfeasible by removing some sets in T_j; more precisely $p(D) \geq P$ and $P < p(D) + \Delta(T_c, D)$.

At a very high level, the intuition behind the INCREASE/DECREASE scheme is as follows. In each call one of three things must occur:

(i) A feasible cover with a small coverage excess is found (Lines 2-3), or
(ii) The call is propagated down the tree at no cost (Lines 4-5), or
(iii) A subtree T_j is split (Lines 6-9). In this case, the cost c_j cannot be accounted for, but the offset in coverage $|P - p(D)|$ is reduced at least by a factor of 3.

If the INCREASE/DECREASE algorithms split many subtrees (incurring a high extra cost) then the offset in coverage must have been very high at the beginning, which means the cost of the dual solution is high and so the splitting cost can be charged to it. In order to flesh out these ideas into a formal proof we need to establish some crucial properties of the merger graph and the algorithms. Proofs are omitted due to lack of space.

Lemma 4.5. *If $y_i < \lambda\, p_i$ then there exist $j' \in \widehat{C}$ and $j'' \in \widehat{C}^-$ such that either $j' = j''$ or $(j', j'') \in E$ or $(j'', j') \in E$.*

Lemma 4.6. *Let (j, D) be the input of* INCREASE/DECREASE. *Then at the beginning of each call we have $j' \in D$ or $j'' \in D$ for all $(j', j'') \in E$. Furthermore, if $j' \in D$ and $j'' \in D$ then j' or j'' must have been split in a previous call.*

Lemma 4.7. *Let (j, D) be the input of* INCREASE/DECREASE. *For* INCREASE *we always have $p(D) \le P < p(D) + \Delta(T_j, D)$, and for* DECREASE *we have $p(D) \ge P > p(D) + \Delta(T_j, D)$.*

Recall that y is also a feasible solution for the dual relaxation of P-TBC and its cost is given by $\mathrm{DL} = \sum_{i=1}^{n} y_i - (p(U) - P)\lambda$. The following lemma proves the upper bound of Theorem 4.1.

Lemma 4.8. *Suppose* MERGE *outputs D. Then $c(D) \le \big(1 + \frac{1}{3^{k-1}}\big)\mathrm{DL} + k\, c_{\max}$ for all $k \in Z_+$.*

Proof. Let us digress for a moment for the sake of exposition. Suppose that in Line 6 of MERGE, instead of calling INCREASE, we return $D' = D \oplus T_r$. Notice every arc in the merger graph has exactly one endpoint in D'. By Lemma 4.5, any element i not covered by D' must have $y_i = \lambda\, p_i$. Furthermore, if $y_i > 0$ then there exists at most one set in D' that covers i; if two such sets exist, one must dominate the other in y and y^-, which is not possible. Hence,

$$c(D) = \sum_{j \in D'} \sum_{i : a_{ij} = 1} y_i = \sum_{\substack{i \text{ s.t.} \\ \exists\, j \in D' : a_{ij} = 1}} y_i = \sum_{i \in U} y_i - (p(U) - p(D'))\lambda \le \mathrm{DL} + (p(D') - P)\lambda \quad (4.7)$$

In the fortunate case that $(p(D') - P)\lambda \le k c_{\max}$, the lemma would follow. Of course, this need not happen and this is why we make the call to INCREASE instead of returning D'.

Let j_q be the root of the q^{th} subtree split by INCREASE/DECREASE. Also let D_q the solution right before splitting T_{j_q}, and D'_q and D''_q be the unfeasible/feasible pair of solutions after the splitting, which are used as parameters in the recursive calls (Lines 11-12). Suppose Lines 7-9 processed only one child of j_q, this can only happen in INCREASE, in which case $p(D''_q) > P$ but $p(D''_q) - b_{j_q} < P$. The same argument used to derive (4.7) gives us

$$c\left(D''_q \setminus \{j_{\le q}\}\right) \le \sum_{i \in U} y_i - \left(p(U) - p(D''_q) + b_{j_q}\right)\lambda \le \mathrm{DL} \quad (4.8)$$

The cost of the missing sets is $c(\{j_{\le q}\}) \le q\, c_{\max}$, thus if $q \le k$ the lemma follows. A similar bound can be derived if the recursive call ends in Line 3 before splitting the k^{th} subtree.

Finally, the last case to consider is when Lines 7-9 process two or more children j_q for all $q \leq k$. In this case

$$|p(D_q) - P| \geq 3 \min\left\{|p(D'_q) - P|, |p(D''_q) - P|\right\} = 3\,|p(D_{q+1}) - P|, \tag{4.9}$$

which implies $|p(D_1) - P| \geq 3^{k-1}|p(D_k) - P| \geq 3^{k-1}|p(D''_k) - P|$. Also, $\lambda(P - p(D_1)) \leq \mathrm{DL}$ since all elements i not covered by D_1 must be such that $y_i = \lambda p_i$. Hence, as before

$$c\left(D''_k \setminus \{j_{\leq k}\}\right) \leq \mathrm{DL} + \left(p(D''_k) - P\right)\lambda \leq \mathrm{DL} + \frac{P - p(D_1)}{3^{k-1}} \leq \left(1 + \frac{1}{3^{k-1}}\right)\mathrm{DL} \tag{4.10}$$

Adding the cost of $\{j_{\leq k}\}$ we get the lemma. ∎

5. Concluding remarks and open problems

The results in this paper suggest that Lagrangian relaxation is a powerful technique for designing approximation algorithms for partial covering problems, even though the black-box approach may not be able to fully realize its potential.

It would be interesting to extend this study on the strengths and limitation of Lagrangian relaxation to other problems. The obvious candidate is the k-Median problem. Jain and Vazirani [21] designed a 2α-approximation for k-Median using as a black box an α-LMP approximation for Facility Location. Later, Jain et al. [22] gave a 2-LMP approximation for Facility Location. Is the algorithm in [21] optimal in the sense of Theorem 3.1? Can the algorithm in [22] be turned into a 2-approximation for k-Median by exploiting structural similarities when combining the two solutions?

Acknowledgments: I am indebted to Danny Segev for sharing an early draft of [26] and for pointing out Kolen's work. Also thanks to Mohit Singh and Arie Tamir for helpful discussions and to Elena Zotenko for suggesting deriving the result of Section 3.

References

[1] S. Arora and G. Karakostas. A $2+\epsilon$ approximation algorithm for the k-MST problem. In *Proc., 11th Symposium on Discrete Algorithms*, pp. 754–759, 2000.

[2] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for approximating the weighted vertex cover. *Journal of Algorithms*, 2:198–203, 1981.

[3] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[4] C. Berge. Balanced matrices. *Mathematical Programming*, 2:19–31, 1972.

[5] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proc., Symp. on Foundations of Computer Science*, pp. 378–388, 1999.

[6] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc., 12th Symposium on Discrete Algorithms*, pp. 642–651, 2001.

[7] F. A. Chudak, T. Roughgarden, and D. P. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. In *Proceedings of the 9th Integer Programming and Combinatorial Optimization Conference (IPCO)*, pp. 60–70, 2001.

[8] K. L. Clarkson. A modification of the greedy algorithm for vertex cover. *Information Processing Letters*, 16(1):23–25, 1983.

[9] U. Feige. A threshold of ln n for approximating set cover. *J. of the ACM*, 45(4):634–652, 1998.

[10] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[11] N. Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proc., 37th Symposium on Theory of computing*, pp. 396–402, 2005.
[12] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proc., 37hd Symposium on Foundations of Computer Science*, pp. 302–309, 1996.
[13] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
[14] D. R. Gaur, T. Ibaraki, and R. Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. *Journal of Algorithms*, 43 (1):138–152, 2002.
[15] D. Golovin, V. Nagarajan, and M. Sing. Approximating the k-multicut problem. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.
[16] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.
[17] R. Hassin and D. Segev. Rounding to an integral program. In *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA'05)*, pp. 44–54, 2005.
[18] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.
[19] D. S. Hochbaum, editor. *Approximation Algorithms for NP–hard Problems*. PWS Publishing Company, 1997.
[20] A. J. Hoffman, A. Kolen, and M. Sakarovitch. Totally-balanced and greedy matrices. *SIAM Journal on Algebraic and Discrete Methods*, 6:721–730, 1985.
[21] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2): 274–296, 2001.
[22] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6): 795–824, 2003.
[23] G. Karakostas. A better approximation ratio for the vertex cover problem. In *Proc., 15th International Colloquium on Automata, Languages, and Programming*, pp. 1043–1050, 2005.
[24] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Plenum Press, 1972.
[25] A. Kolen. *Location problems on trees and in the rectilinear plane*. PhD thesis, Matematisch Centrum, Amsterdam, 1982.
[26] J. Könemann, O. Parekh, and D. Segev. A unified approach to approximating partial covering problems. In *Proc. 14th European Symposium on Algorithms (ESA)*, pp. 468–479, 2006.
[27] J. Könemman and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM Journal of Computing*, 31:1783–1793, 2002.
[28] A. Levin and D. Segev. Partial multicuts in trees. In *Proc. 3rd Intern. Workshop on Approx. and Online Algorithms (WAOA)*, 2005.
[29] N. Megiddo. Combinatorial optimization with rational objective functions. In *Proceedings of the tenth annual ACM symposium on Theory of computing (STOC)*, pp. 1–12, 1978.
[30] O. Parekh and D. Segev. Path hitting in acyclic graphs. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pp. 564–575, 2006.
[31] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem (extended abstract). In *5th Scandinavian Workshop on Algorithm Theory (SWAT)*, pp. 66–75, 1996.
[32] J. P. Spinard. Doubly lexical ordering of dense 0-1 matrices. *Information Processing Letters*, 45:229–235, 1993.