On the Equivalence Between the Primal-Dual Schema and the Local-Ratio Technique

Reuven Bar-Yehuda and Dror Rawitz

Department of Computer Science, Technion, Haifa 32000, Israel
{reuven,rawitz}@cs.technion.ac.il

Abstract. We discuss two approximation approaches, the primal-dual schema and the local-ratio technique. We present two relatively simple frameworks, one for each approach, which extend known frameworks for covering problems. We show that the two are equivalent, and conclude that the integrality gap of an integer program serves as a bound to the approximation ratio when working with the local-ratio technique.

1 Introduction

The *primal-dual method* for solving linear programs was originally proposed by Dantzig, Ford, and Fulkerson [10] for solving linear programs. Over the years this method has become an important tool for solving combinatorial optimization problems. Many algorithms which solve combinatorial optimization problems such as Dijkstra's *shortest path* algorithm [11], Ford and Fulkerson's *network flow* (or *minimum* s,t*-cut*) algorithm [13], and Kuhn's assignment algorithm [20] either use the method or can be analyzed by it (see, e.g., [15] for more details).

Combinatorial optimization problems can often be formulated as integer programs. However, unlike the above mentioned problems, the LP-relaxation of many problems have non-integral optimal solutions (and an *integrality gap* which is greater than 1), therefore, the primal-dual method cannot be used to solve them. The *primal dual schema* for approximation is a modified version of the primal-dual method (see [15]). While both complementary slackness conditions are imposed in the classical setting, we impose the primal conditions but relax the dual conditions when working with the primal-dual schema. A primal-dual approximation algorithm constructs an approximate primal solution and a feasible dual solution simultaneously. The approximation ratio is derived from comparing the values of both solutions. Many LP-based algorithms can be analyzed by the primal-dual schema. However, the first algorithm using this schema is Bar-Yehuda and Even's [4] approximation algorithm for the *vertex cover* problem (and the *set cover* problem). Following the work of Goemans and Williamson [14], Bertsimas and Teo [8] proposed a primal-dual framework to design and analyze approximation algorithms for integer programming problems of the covering type. As in [14] this framework enforces the primal complementary slackness conditions while relaxing the dual conditions. A detailed survey on the primal dual schema was written by Goemans and Williamson [15].

Recently, Jain and Vazirani [19] presented a primal-dual approximation algorithm which uses a different approach from the one described here. Instead of the "usual" mechanism of relaxing the complementary slackness conditions, they relaxed the primal conditions. As far as we know, this is the only primal-dual algorithm which works with programs that have negative coefficients.

The local-ratio approach was developed by Bar-Yehuda and Even [5] in order to approximate the *weighted vertex cover* problem. Ten years later Bafna, et al. [1] extended the *Local-Ratio Theorem* from [5] by introducing the idea of minimal covers in order to construct a 2-approximation algorithm for the *feedback vertex set* problem. Later, Bar-Yehuda [3] presented a unified local-ratio approach for developing and analyzing approximation algorithms for covering problems. He further extended the Bafna et al. [1] extension of the Local-Ratio Theorem. His extension yields a generic r-approximation algorithm, which can explain most known optimization and approximation algorithms for covering problems. Lately, Bar-Noy et al. [2] have presented an approximation framework for solving resource allocation and scheduling problems. Their algorithms are based on the local-ratio technique, but can be interpreted within the primal-dual schema. This study was the first to present a primal-dual or a local-ratio approximation algorithm for a natural maximization problem.

We present a local-ratio framework which generalizes the one from [3], in which only non-negative weight subtractions were considered. We found this limitation to be redundant, and, therefore, our generic algorithm can be used to design and analyze a wider family of algorithms. We also present a primal-dual approximation framework which extends the generic algorithm from [8]. As done before, our framework relaxes the dual complementary slackness conditions. However, the use of valid inequalities with non-negative coefficients enables us to widen the variety of algorithms which fall with in the scope of our framework. Furthermore, due to [2], we are able to extend our frameworks to handle maximization problems. (This extension will appear in the full version.) By showing that primal-dual's valid inequalities and local-ratio's weight functions are equivalent notions we show that both frameworks are actually one and the same. A corollary to this equivalence is that the *integrality gap* of an integer program serves as a bound to the approximation ratio of a local-ratio algorithm (as in the case of a primal-dual algorithm). Also, note that in cases where a primal-dual algorithm serves as a subroutine (e.g., multyphase primal-dual [21], see also [8]) it can be replaced by the corresponding local-ratio algorithm. We demonstrate the combined approach on a variety of problems. First, we present a linear time approximation algorithm for the *generalized hitting set* problem. This algorithm achieves a ratio of 2 in the special case of the *generalized vertex cover* problem, and is better in time complexity than Hochbaum's [18] $O(nm \log \frac{n^2}{m})$ 2-approximation algorithm for this special case. We exhibit the first local-ratio algorithm which uses negative weights by presenting an analysis to Ford and Fulkerson's [13] algorithm for minimum s,t-cut. Furthermore, in order to explain this algorithm within the primal-dual schema, we use a linear program with inequalities which have some negative coefficients. We also analyze

2-approximation algorithms for the *minimum feedback vertex set* problem [7], and for a non-covering problem called the *minimum 2-satisfiability* problem [16, 6]. Finally, we explain a $\frac{1}{2}$-approximation algorithm for a maximization problem called the *interval scheduling* problem [2]. This example provides some insight to the design and analysis of approximation algorithms for maximization problems, and to the equivalence between the two approaches in the maximization case.

We believe that this study contributes to the understanding of both approaches, and, especially, that it will help in the design of algorithms for maximization problems. Another point of interest is that while the use of a linear objective function and linear advancement steps are an integral part of both techniques, our frameworks are not limited to linear integer programs (though we did not find a natural non-linear application).

2 Definitions

We consider the following problem: Given a non-negative *penalty* (or *profit*) vector $p \in \mathbb{R}^n$, find a solution x that minimizes (or maximizes) the inner product $p \cdot x$ subject to some set $\mathcal{F}$ of feasibility constraints (not necessarily linear) on $x \in \{0,1\}^n$. Note that we will sometimes abuse the notation by treating a vector $x \in \{0,1\}^n$ as the set of its 1 coordinates, i.e., as $\{j : x_j = 1\}$. The correct interpretation should be clear from the context.

A minimization (maximization) problem $(\mathcal{F}, p)$ is called a *covering* (*packing*) problem if $\mathcal{F}$ is monotone, i.e., for all x' such that $x' \geq x$ ($x' \leq x$) if x satisfies $\mathcal{F}$ then x' satisfies $\mathcal{F}$. Note that a monotone set of linear constraints typically includes inequalities with non-negative coefficients.

Primal-dual approximation algorithms for covering problems traditionally reduce the size of the problem at hand in each iteration by adding an element whose corresponding dual constraint is tight to the primal solution (see [15, 8]). Local-ratio algorithms for covering problems implicitly add all zero penalty elements to the solution, and, therefore, reduce the size of the problem in each step as well (see [3]). In order to implement this we alter the problem definition by adding a set (or vector), denoted by z, which includes elements which are considered to be taken into the solution. This makes it easier to present primal-dual algorithms recursively, and to present local-ratio algorithms in which the addition of zero penalty elements to the partial solution is explicit.

More formally, given a minimization (maximization) problem $(\mathcal{F}, p)$ and a vector z, we are interested in the following problem: Find a vector x such that (1) $z \cap x = \emptyset$; (2) $x \cup z$ satisfies $\mathcal{F}$ (x in the maximization case); And, (3) minimizes (maximizes) the inner product $p \cdot x$. (Note that when $z = \emptyset$ we get the original problem.) We define the following for a minimization problem $(\mathcal{F}, p, z)$. A vector x is called a *feasible solution* if $z \cap x = \emptyset$ and $z \cup x$ satisfies $\mathcal{F}$. We denote the set of feasible solutions with respect to $\mathcal{F}$ and z by $S(\mathcal{F}, z)$. A feasible solution x^* is called an *optimal solution* if every feasible solution x satisfies $p \cdot x^* \leq p \cdot x$. A feasible solution x is called an *r-approximation* if $p \cdot x \leq r \cdot p \cdot x^*$, where x^* is an optimal solution. The corresponding definitions for a maximization problem can be understood in a strait-forward manner. Given a minimization (maximization) problem $(\mathcal{F}, p, z)$, a feasible solution x is called *minimal* (*maximal*) if for all $j \in x$ ($j \notin x \cup z$) the vector $x \setminus \{j\}$ ($x \cup \{j\}$) is not feasible.

3 Local-Ratio

The following is the Local-Ratio Theorem given in our terminology.

Theorem 1 ([2]). *Let $\mathcal{F}$ be a set of constraints, and let z be a boolean vector. Also, let p, p_1, and p_2 be penalty (or profit) vectors such that $p = p_1 + p_2$. Then, if x is an r-approximation with respect to $(\mathcal{F}, p_1, z)$ and $(\mathcal{F}, p_2, z)$, then x is an r-approximation with respect to $(\mathcal{F}, p, z)$.*

Note that $\mathcal{F}$ can include arbitrary feasibility constraints and not just linear, or linear integer, constraints. Nevertheless, all successful applications of the local ratio technique to date involve problems in which the constraints are linear.

Bar-Yehuda [3] defined the following for covering problems:

Definition 1. *Given a covering problem $(\mathcal{F}, p)$, a weight function δ is called* r-effective *with respect to $\mathcal{F}$, if every minimal solution x with respect to $(\mathcal{F}, \delta)$ satisfies $\delta x \le r \cdot \delta x^*$, where x^* is an optimal solution.*

We prefer the following more practical definition:

Definition 2. *Given a minimization (maximization) problem $(\mathcal{F}, p, z)$, a weight function δ is called* r-effective with respect to $(\mathcal{F}, z)$, *if $\forall j \in z, \delta_j = 0$, and there exists β such that every minimal (maximal) solution x with respect to $(\mathcal{F}, z)$ satisfies: $\beta \le \delta \cdot x \le r\beta$ ($r\beta \le \delta \cdot x \le \beta$). In this case we say that β is a* witness *to δ's r-effectiveness. If $z = \emptyset$ we say that δ is* r-effective with respect to $\mathcal{F}$.

Obviously, by assigning $\beta = \delta x^*$, where x^* is an optimal solution, we get that the first definition implies the latter. For the other direction, note that $\beta \le \delta x^*$.

A local-ratio algorithm for a covering problem works as follows. First, construct an r-effective weight function δ, such that $p - \delta \ge 0$ and there exists some j for which $p_j = \delta_j$. Such a weight function is called *p-tight*. Subtract δ from the penalty vector p. Add all zero penalty elements to the partial solution z. Then, recursively solve the problem with respect to $(\mathcal{F}, p - \delta, z)$. When the empty set becomes feasible (or, when z becomes feasible with respect to $\mathcal{F}$) the recursion terminates. Finally, remove unnecessary elements from the temporary solution by performing a reverse deletion phase. Algorithm LRcov is a modified version of the generic local-ratio approximation algorithm for covering problems from [3]. (The initial call is LRcov$(p, \emptyset)$.) The main difference between the algorithm from [3] and the one given here is that in the latter the augmentation of the temporary solution is done one element at a time. By doing this we have the option not to include zero penalty elements which do not contribute to the feasibility of the partial solution z.

Algorithm LRcov(p, z)

1. If $\emptyset \in S(\mathcal{F}, z)$ return $\emptyset$
2. Construct a p-tight weight function δ which is r-effective w.r.t. $(\mathcal{F}, z)$
3. Let $j \notin z$ be an index for which $\delta_j = p_j$
4. $x \leftarrow \text{LRcov}(p - \delta, z \cup \{j\})$
5. If $x \notin S(\mathcal{F}, z)$ then $x \leftarrow x \cup \{j\}$
6. Return x

Theorem 2. *Algorithm LRcov outputs an r-approximate solution.*

Proof. Let p^k, δ^k and z^k be p, δ, and z, respectively, at the end of the k'th level, and let t be the maximal depth of the recursion. We prove by reverse induction on k that Algorithm LRcov returns an minimal r-approximation with respect to $(\mathcal{F}, p^k, z^k)$. For $k = t$, $\emptyset$ is a minimal optimal solution. Otherwise, examine x at the end of the k'th level. By the induction hypothesis $x \setminus \{j\}$ is an minimal r-approximation with respect to $(\mathcal{F}, p^{k+1}, z^{k+1})$. Therefore, x is a minimal solution with respect to $(\mathcal{F}, z^k)$. Moreover, due to the fact that $p_j^{k+1} = 0$ x is an r-approximation with respect to $(\mathcal{F}, p^{k+1}, z^k)$. By the r-effectiveness of δ^k and Theorem 1 x is an r-approximate solution with respect to $(\mathcal{F}, p^k, z^k)$.

Algorithm LRcov finds an r-effective weight function for the current problem, adds a zero penalty element to the solution, and then changes the problem. Such an algorithm heavily relies upon the fact that the constraint set $\mathcal{F}$ is monotone, and, therefore, the optimum of $(\mathcal{F}, p^{k+1}, z^{k+1})$ is not greater than the optimum of $(\mathcal{F}, p^{k+1}, z^k)$. Algorithm LRmin uses a slightly different approach. Instead of using r-effective weight functions with respect to $(\mathcal{F}, z)$, it uses weight functions which are r-effective with respect to $\mathcal{F}$, and updates p until a minimal zero-penalty solution is found. By using this "subtract first, ask questions later" approach we are able to approximate problems by using inequalities with some negative coefficients or weight functions with some negative weights. We illustrate this approach in the Sect. 6.

Algorithm LRmin(p)

1. Search for a feasible minimal solution $x \subseteq \{j : p_j = 0\}$
2. If such a solution was found return x
3. Construct a p-tight weight function δ which is r-effective w.r.t. $\mathcal{F}$
4. $x \leftarrow \text{LRmin}(p - \delta)$
5. Return x

Theorem 3. *Algorithm LRmin outputs an r-approximate solution.*

4 Primal-Dual

In [15] Goemans and Williamson presented a generic algorithm based on the *hitting set* problem which is defined as follows: Given subsets $T_1, \ldots, T_q \subseteq E$ and a non-negative cost c_e for every element $e \in E$, find a minimum-cost subset

$x \subseteq E$ such that $x \cap T_i \neq \emptyset$ for every $i \in \{1, \ldots, q\}$. In turns out that many known problems are special cases of this problem. The LP-relaxation of its formulation as an integer program and the corresponding dual are:

$$\begin{array}{llll} \min \sum_{e \in E} c_e x_e & & \qquad \max \sum_{i=1}^{q} y_i & \\ \text{s.t. } \sum_{e \in T_i} x_e \geq 1 & \forall i \in \{1, \ldots, q\} & \qquad \text{s.t. } \sum_{i: e \in T_i} y_i \leq c_e & \forall e \in E \\ \quad x_e \geq 0 & \forall e \in E & \qquad \quad y_i \geq 0 & \forall i \in \{1, \ldots, q\} \end{array}$$

where $x_e = 1$ iff $e \in x$. The primal complementary slackness conditions are: $x_e > 0 \Longrightarrow \sum_{i: e \in T_i} y_i = c_e$, and the dual conditions are: $y_i > 0 \Longrightarrow |x \cap T_i| = 1$.

The algorithm from [15] starts with the dual solution $y = 0$ and the non-feasible primal solution $x = \emptyset$. Then it iteratively increases the both solutions until the primal solution becomes feasible. As in the primal-dual method, if we cannot find a feasible primal solution, then there is a way to increase the dual solution. In this case, if x is not feasible there exists a k such that $x \cap T_k = \emptyset$. Such a set is referred to as *violated*. In each iteration a *violation oracle* supplies a collection of violated subsets $\mathcal{V} \subseteq \{T_1, \ldots, T_q\}$[1]. Then we increase *simultaneously and at the same speed* the dual variables corresponding to subsets in $\mathcal{V}$. When x becomes feasible a *reverse delete step* is performed. This step discards as many elements as possible from the primal solution x without losing feasibility.

Let x^f and ϵ_j be the set output by the algorithm, and the increase of the dual variables corresponding to $\mathcal{V}_j$ in iteration j. Then, $y_i = \sum_{j: T_i \in \mathcal{V}_j} \epsilon_j$, $\sum_{i=1}^{p} y_i = \sum_{j=1}^{l} |\mathcal{V}_j| \epsilon_j$, and $c(x^f) = \sum_{e \in x_f} c_e = \sum_{i=1}^{q} \left|x^f \cap T_i\right| y_i = \sum_{i=1}^{q} \left|x^f \cap T_i\right| \sum_{j: T_i \in \mathcal{V}_j} \epsilon_j = \sum_{j=1}^{l} \epsilon_j \sum_{T_i \in \mathcal{V}_j} \left|x^f \cap T_i\right|$. From this, it is clear that the cost of x^f is at most the value of the dual solution times r if $\forall j \in \{1, \ldots, l\}$ $\sum_{T_i \in \mathcal{V}_j} \left|x^f \cap T_i\right| \leq r \left|\mathcal{V}_j\right|$. Examine iteration j of the reverse deletion step. We know that when e_j was considered for removal, no element $e_{j'}$ with $j' < j$ has been already removed. Thus, after e_j is considered for removal the temporary solution is $x^j = x^f \cup \{e_1, \ldots, e_{j-1}\}$. x^j is called a *minimal augmentation* of $\{e_1, \ldots, e_{j-1}\}$, i.e., x^j is feasible and $x^j \setminus \{e\}$ is not feasible for all $e \in x^j \setminus \{e_1, \ldots, e_{j-1}\}$. Moreover, $\sum_{T_i \in \mathcal{V}_j} \left|x^f \cap T_i\right| \leq \sum_{T_i \in \mathcal{V}_j} \left|x^j \cap T_i\right|$. Therefore, to achieve the above bound Goemans and Williamson have set the following requirement: $\sum_{T_i \in \mathcal{V}_j} |x' \cap T_i| \leq r \left|\mathcal{V}_j\right|$ for every $j \in \{1, \ldots, l\}$ and for every minimal augmentation x' of $\{e_1, \ldots, e_{j-1}\}$. We formalized this by the following.

Definition 3. *A set $\mathcal{V} \subseteq \{T_1, \ldots, T_q\}$ is called* r-effective with respect to $(\mathcal{F}, z)$, *if $\sum_{T_i \in \mathcal{V}} |x \cap T_i| \leq r |\mathcal{V}|$ for every minimal solution with respect to $(\mathcal{F}, z)$, x.*

As did Bertsimas and Teo [8] we prefer to speak in terms of inequalities.

Definition 4. *A set of valid inequalities $\left\{\alpha^1 x \geq \beta^1, \ldots, \alpha^k x \geq \beta^k\right\}$ (or $\leq$ in the maximization case) is called* r-effective with respect to $(\mathcal{F}, p, z)$, *if $\alpha^k_j = 0$ for all k and $j \in z$, and every integral minimal feasible solution with respect to $(\mathcal{F}, z)$, x, satisfies: $\sum_{i=1}^{k} \alpha^i x \leq r \sum_{i=1}^{k} \beta^i$. ($\geq$ in the maximization case.) If the set contains a single inequality we will refer to this inequality as r-effective.*

[1] We could allow the oracle to return some sets which are not violated. See [15].

An r-effective collection $\mathcal{V}$ can be understood as the following r-effective set of valid inequalities: $\left\{\sum_{e\in T_i} x_e \geq 1 : T_i \in \mathcal{V}\right\}$. On the other hand, the latter definition allows the use of other kinds of inequalities (e.g., Sect. 6). Thus, our goal is to find a set of r-effective valid inequalities in order to increase the dual solution. However, it is enough to construct a single r-effective valid inequality for that purpose. This is because a set of valid inequalities $\left\{\alpha^i x \geq \beta^i\right\}_{i=1}^k$ satisfies $\sum_{i=1}^k \alpha^i x \leq r\sum_{i=1}^k \beta^i$ by definition. On the other hand, $\forall i, \alpha^i x \geq \beta^i$, and, therefore, $\sum_{i=1}^k \alpha^i x \geq \sum_{i=1}^k \beta^i$. Thus, we have found an r-effective inequality: $\sum_{j=1}^n (\sum_{i=1}^k \alpha^i)_j x_j \geq \sum_{i=1}^k \beta^i$.

Bertsimas and Teo [8] presented a primal-dual algorithm for linear covering problems which utilizes a single valid inequality in each iteration. This algorithm constructs a valid inequality in each iteration, and uses it to modify the current instance. After this modification at least one of the coordinates of the penalty vector p becomes zero, and this makes it possible to reduce the size of the problem in each iteration. Thus, the algorithm terminates after no more than n iterations. The performance of this algorithm depends on the choice of the inequalities. In fact, it corresponds to what Bertsimas and Teo call the *strength* of an inequality, which is the minimal r for which it is r-effective.

Algorithm PDcov is a recursive version of the algorithm from [8]. (The initial call is PDcov$(p, \emptyset, 1)$.) Informally, it can be viewed as follows: construct an r-effective inequality; update the corresponding dual variable and the vector p such that p remains non-negative; find an element j whose penalty is zero; add j to the temporary partial solution z; then recursively solve the problem with respect to $\mathcal{F}, z$ and the new penalty vector; the termination condition of the recursion is met when the empty set becomes a feasible solution; finally, a reverse deletion phase removes unnecessary elements from the temporary solution. There are two differences between the algorithm from [8] and our algorithm. First, we present our algorithm recursively, and, more importantly, our algorithm is not restricted to linear integer programs.

Algorithm PDcov(p, z, k)

1. If $\emptyset \in S(\mathcal{F}, z)$ return $\emptyset$
2. Construct a valid inequality $\alpha^k x \geq \beta^k$ which is r-effective w.r.t. $(\mathcal{F}, z)$
3. $y_k \leftarrow \max\left\{\epsilon : p - \epsilon\alpha^k \geq 0\right\}$
4. Let $j \notin z$ be an index for which $p_j = y_k \alpha_j^k$
5. $x \leftarrow$ PDcov$(p - y_k\alpha^k, z \cup \{j\}, k+1)$
6. If $x \notin S(\mathcal{F}, z)$ then $x \leftarrow x \cup \{j\}$
7. Return x

The following theorem is based on the corresponding theorem from [8].

Theorem 4. *Algorithm PDcov outputs an r-approximate solution.*

Proof. Denote the maximum depth of the recursion by t, and let z^k be the temporary partial solution of depth k. Examine the k'th inequality $\alpha^k x \geq \beta^k$ for some k. This is a valid inequality with respect to $(\mathcal{F}, z^k)$, and, therefore, it is valid with respect to $\mathcal{F}$. Thus, LP2 $= \min\left\{px : \forall k \in \{1,\ldots,t\}, \alpha^k x \geq \beta^k \text{ and } x \geq 0\right\}$ is a relaxation of $(\mathcal{F}, p)$. By the construction of x, it is a feasible solution for $\mathcal{F}$, and, therefore, for LP2. Also, y is a feasible solution for the dual of LP2.

Let x^k be the value returned by the k'th call to the recursion, let p^k be the penalty vector of depth k, and let $j(k)$ be the chosen element in the k'th call. We prove by induction that $p^k x^k \leq r \sum_{l \geq k} y_l b^l$. First, for $k = t$, we have $p^t x^t = 0$. For $k < t$ we have,

$$p^k x^k = (p^{k+1} + y_k \alpha^k) x^k \overset{(1)}{=} p^{k+1} x^{k+1} + y_k \alpha^k x^k \overset{(2)}{\leq} r \sum_{l \geq k+1} y_l \beta^l + y_k r \beta^k = r \sum_{l \geq k} y_l \beta^l$$

where (1) is because $p^{k+1}_{j(k)} = 0$, and (2) is implied by the induction hypothesis and the r-effectiveness of the inequality. Finally, $px = p^1 x^1 \leq r \sum_{l \geq 1} y_l \beta^l \leq r \cdot \mathrm{Opt}(\mathrm{LP2}) \leq r \cdot \mathrm{Opt}(\mathcal{F}, p)$ and, therefore, x is an r-approximate solution.

Algorithm PDmin uses inequalities which are r-effective with respect to $\mathcal{F}$, instead of using inequalities which are r-effective with respect to $(\mathcal{F}, z)$ (as in the local-ratio case). It uses such inequalities, while updating p, and the dual solution, until a minimal zero-penalty solution is found.

Algorithm PDmin(p)

1. Search for a feasible minimal solution $x \subseteq \{j : p_j = 0\}$
2. If such a solution was found return x
3. Construct a valid inequality $a^l x \geq b_l$ which is r-effective w.r.t. to $\mathcal{F}$
4. $y_l \leftarrow \max\left\{\epsilon : p - \epsilon a^l \geq 0\right\}$
5. $x \leftarrow$ PDmin2$(p - y_l a^l)$
6. Return x

Theorem 5. *Algorithm PDmin outputs an r-approximate solution.*

5 Equivalence

Lemma 1. *$\alpha x \geq \beta$ is an r-effective inequality iff α is an r-effective weight function with the witness β.*

Proof. We prove the lemma for minimization problems. Similar arguments can be used in the maximization case. Let $\alpha x \geq \beta$ be an r-effective inequality. By definition every minimal feasible solution x satisfies: $\beta \leq \alpha x \leq r\beta$. Thus, α is an r-effective weight function. On the other hand, let α be an r-effective weight function with a witness β. Due to the r-effectiveness of α every minimal feasible solution x satisfies $\beta \leq \alpha x \leq r\beta$. Therefore, $\alpha x \geq \beta$ is an r-effective inequality.

By Lemma 1 the use of valid inequalities can be explained by utilizing the corresponding weight functions and vice versa. Thus, Algorithms PDcov and LRcov are actually identical. The same goes for Algorithms PDmin and LRmin. From this we can also conclude that the integrality gap of an integer program serves as a bound to the approximation ratio of a local-ratio algorithm which can be explained by algorithms LRcov or LRmin.

6 Examples

Generalized Hitting Set. The *generalized hitting set* problem is defined as follows. Given a collection of subsets S of a ground set U, a non-negative penalty p_s for every set $s \in S$, and a non-negative penalty p_u for every element $u \in U$, find a minimum-penalty collection of objects $C \subseteq U \cup S$, such that for all $s \in S$, either there exists $u \in C$ such that $u \in s$, or $s \in C$. As in the *hitting set* problem our objective is to cover the sets in S by using elements from U. However, in this case, we are allowed not to cover a set s, provided we pay a tax p_s. The hitting set problem is the special case where the tax is infinite for all sets. The generalized hitting set problem can be formalized as follows:

$$\begin{array}{lll} \min & \sum_{u \in U} p_u x_u + \sum_{s \in S} p_s x_s & \\ \text{s.t.} & \sum_{u \in s} x_u + x_s \geq 1 & \forall s \in S \\ & x_t \in \{0,1\} & \forall t \in U \cup S \end{array}$$

Observe that paying the penalty p_s is required only when s is not covered. Thus, $\sum_{u \in s} x_u + x_s \geq 1$ is Δ_S-effective for any set $s \in S$, where $\Delta_S = \max\{|s| : s \in S\}$. The corresponding Δ_S-effective weight function would be: $\delta_s(t) = \epsilon$ if $t \in \{s\} \cup s$, and $\delta_s(t) = 0$ otherwise. The inequality remains Δ_S-effective if we use any value between 1 and Δ_S as x_s's coefficient. Analogously, $\delta_s(s)$ can take any value between ϵ and $\Delta_S \epsilon$. We can approximate this problem by using the "standard" approach for covering problems (when using the above inequalities or weight functions). However, we can also use a PDmin (or LRmin) to construct a linear time Δ_S-approximation algorithm. First, we use all the above inequalities in an arbitrary order; then a zero penalty minimal feasible solution can be found: all zero penalty elements and all the sets which are not covered by some zero penalty element. This would be a Δ_S-approximate solution.

Feedback Vertex Set. Let $G = (V, E)$ be an undirected graph, and let p be a penalty vector. The *feedback vertex set* problem is to find a minimum penalty set $F \subseteq V$ whose removal from G leaves an acyclic graph. Becker and Geiger [7] and Chudak et al. [9] proved that $\sum_{v \in V} \deg(v) \leq 2 \sum_{v \in F^*} \deg(v)$ for every minimal feedback vertex set F, where F^* is an optimal solution. Bar-Yehuda [3] indicated that this actually means that the weight function $\delta(v) = \deg(v)$ is 2-effective, and then used this weight function to simplify the presentation of Becker and Geiger's [7] 2-approximation algorithm. The corresponding 2-effective inequality is $\sum_{v \in V} \deg(v) x_v \geq \sum_{v \in F^*} \deg(v)$. Therefore, a primal-dual analysis to this algorithm can be given by using algorithm PDcov. It is important to note that you do not need to know the value $\sum_{v \in F^*} \deg(v)$ in order to execute the algorithm.

Minimum s,t-cut. Given an undirected graph $G = (V, E)$, a pair of vertices s, t, and a non-negative penalty for each edge $e \in E$, we wish to find a cut of minimum penalty. In order to solve this problem we solve the directed version: given a digraph $G = (V, E)$, a pair of vertices s and t, and a non-negative penalty p_e for each arc $e \in E$, find a directed cut of minimum penalty. Denote by Q the set of all (directed) paths from s to t. The minimum s,t-cut problem can be formalized by using the following integer program:

$$\begin{array}{ll} \min \sum_{e \in E} p_e x_e & \\ \text{s.t. } \sum_{e \in q} x_e \geq 1 & \forall q \in Q \\ \quad x_e \in \{0,1\} & \forall e \in E \end{array}$$

where $x_e = 1$ iff e is in the cut.

We analyze Ford and Fulkerson algorithm [13]. Examine the inequality: $\sum_{e \in q} x_e - \sum_{e \in \overline{q}} x_e \geq 1$ for some path $q \in Q$, where $\overline{q}$ is the corresponding path from t to s (i.e., the one going in the opposite direction). The corresponding weight function is: $\delta_q(e) = (|e \cap q| - |e \cap \overline{q}|) \cdot \epsilon$ for some ϵ. This means that arcs in the 'right' direction are given penalty ϵ, while arcs in the 'wrong' direction are given penalty $-\epsilon$. We show that δ_q and its corresponding inequality are 1-effective. Let F be some minimal feasible solution, i.e, F is a minimal set of arcs which induces an s,t-cut $(S, \overline{S})$. Any path going from s to t must go through F an odd number of times: $k \geq 1$ times from S to $\overline{S}$ and $k-1$ times in the opposite direction. Therefore, $\delta_q(F) = \sum_{e \in F} \delta_e = k \cdot \epsilon + (k-1) \cdot (-\epsilon) = \epsilon$. The algorithm works as follows: iteratively find an unsaturated path q from s to t and subtract δ_q from the p, where ϵ is the value that saturates some arc $e \in q$; when no such path exists pick a minimal set of arcs from the set $\{e : p_e = 0\}$. Note that, in this case, we are allowed to change our minds about an arc, i.e., to increase the penalty of an arc during the execution of the algorithm.

Minimum 2SAT. Given a 2CNF formula φ with m clauses on the variables $x_1, \ldots, x_n$ and a penalty vector p, the *minimum weight 2-satisfiability* problem is to find a minimum penalty truth assignment which satisfies φ (or determine that no such assignment exists). Gusfield and Pitt [16] presented an $O(nm)$ time 2-approximation algorithm for 2SAT. We shall give an intuitive explanation of this algorithm in the context of the approximation frameworks. First, we can check whether φ is satisfiable by using the algorithm from [12]. Therefore, we may assume that φ is satisfiable. In order to construct a 2-approximation algorithm we need to find 2-effective inequalities or weight functions. Given a literal ℓ, let $T(\ell)$ be the set of variables which must be assigned TRUE whenever ℓ is assigned TRUE. Let x_i and x_j be variables such that $x_j \in T(\overline{x_i})$. For such variables the inequality $x_i + x_j \geq 1$ is valid. Moreover, it is easy to see that this inequality is 2-effective. Constructing $T(x_i)$, and a zero penalty feasible solution can be done efficiently by using constraint propagation (for a more detailed analysis see [6]).

Interval Scheduling. Bar-Noy et al. [2] have presented a framework for approximating resource allocation and scheduling problems. We analyze one of the algorithms from [2] which approximates a packing problem called the *interval scheduling* problem. This algorithm does not fall within the frameworks discussed earlier, but the analysis offers some intuition to the equivalence between the two approaches in the maximization case. An interesting point is that while a penalty vector is kept non-negative during the execution of an algorithm, a profit vector is expected to be non-positive when the subtraction phase is over. This means, in primal-dual terms, that in the maximization case the dual solution is initially not feasible, and that it becomes feasible at the end of the algorithm. The negative coordinates of the profit vector correspond to the non-tight constraints.

In the *interval scheduling* problem we are given a set of *activities*, each requiring the utilization of a given *resource*. The activities are specified as a collection of sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$. Each set represents a single activity: it consists of all possible *instances* of that activity. An instance $I \in \mathcal{A}_i$ is defined by the following parameters: (1) A half-open time interval $[s(I), e(I))$ during which the activity will be executed. $s(I)$ and $e(I)$ are called the *start-time* and *end-time* of the instance; And, (2) the profit $p(I) \geq 0$ gained by scheduling this instance of the activity. A *schedule* is a collection of instances. It is feasible if it contains at most one instance of every activity, and at most one instance for all time instants t. In the interval scheduling problem our goal is to find a schedule that maximizes the total profit accrued by instances in the schedule. More formally, our goal is to find an optimal solution to the following integer program on the boolean variables $\{x_I : I \in \mathcal{A}_i, 1 \leq i \leq m\}$.

$$\begin{array}{lll} \max & \sum_I p_I x_I & \\ \text{s.t.} & \sum_{I: s(I) \leq t < e(I)} x_I \leq 1 & \forall t \\ & \sum_{I: I \in \mathcal{A}_i} x_I \leq 1 & \forall i \in \{1, \ldots, m\} \\ & x_I \in \{0, 1\} & \forall i \; \forall I \in A_i, \end{array}$$

The algorithm from [2] works as follows. First, delete all instances with non-positive profit. If no instances remain, return the empty schedule. Otherwise, let J be an instance with minimum end-time, and let $\mathcal{A}(J)$ and $\mathcal{I}(J)$ be the activity to which instance J belongs and the set of instances intersecting J (including J), respectively. Define $\delta(I) = p(J)$ if $I \in \mathcal{A}(J) \cup \mathcal{I}(J)$, and $\delta(I) = 0$ otherwise. Then solve the problem recursively with respect to a new profit vector, $p - \delta$. Let S' be the schedule returned. If $S' \cup \{J\}$ is a feasible solution return $S = S' \cup \{J\}$. Otherwise, return $S = S'$. δ is $\frac{1}{2}$-effective, and, therefore, this is a $\frac{1}{2}$-approximation algorithm. Bar-Noy et al. also presented a primal-dual interpretation to their algorithm. However, in order to do so they modified the original algorithm by using a different $\frac{1}{2}$-effective weight function: $\delta'(J) = p(J)$, $\delta'(I) = \frac{1}{2}p(J)$ if $I \in \mathcal{A}(J) \cup \mathcal{I}(J) \setminus \{J\}$, and $\delta'(I) = 0$ otherwise. We present the following alternative analysis. Consider the inequality $\sum_{I \in \mathcal{I}(J)} x_I + \sum_{I \in A(J)} x_I \leq 2$, which states that only one instance from $\mathcal{A}(J)$ and one from $\mathcal{I}(J)$ can belong to a feasible schedule. This inequality is $\frac{1}{2}$-effective because a maximal schedule contains at least one instance from $\mathcal{A}(J) \cup \mathcal{I}(J)$. The original algorithm can be explained by the $\frac{1}{2}$-effective inequality $\sum_{I \in \mathcal{I}(J) \cup \mathcal{A}(J)} x_I \leq 2$. The difference between δ and δ' (or between their corresponding inequalities) is the ratio between the weight of $\tilde{I}$ and the weights of the other instances. In fact, any value between 1 and 2 would have been acceptable.

Acknowledgments

We thank Eran Mann and especially Ari Freund for helpful discussions.

References

1. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. on Disc. Math.*, 12(3):289–297, 1999.
2. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and scheduling. In *32nd ACM Symposium on the Theory of Computing*, 2000.
3. R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.
4. R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.
5. 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.
6. R. Bar-Yehuda and D. Rawitz. Efficient algorithms for bounded integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001.
7. A. Becker and D. Geiger. Approximation algorithms for the loop cutset problem. In *10th Conference on Uncertainty in Artificial Intelligence*, pages 60–68, 1994.
8. D. Bertsimas and C. Teo. From valid inequalities to heuristics: A unified view of primal-dual approximation algorithms in covering problems. *Operations Research*, 46(4):503–514, 1998.
9. F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal-dual interpretation of recent 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22:111–118, 1998.
10. G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm for linear programs. In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and Related Systems*, pages 171–181. Princeton University Press, 1956.
11. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
12. S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. on Comp.*, 5(4):691–703, 1976.
13. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
14. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. on Comp.*, 24(2):296–317, 1995.
15. M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Hochbaum [17], chapter 4.
16. D. Gusfield and L. Pitt. A bounded approximation for the minimum cost 2-SAT problem. *Algorithmica*, 8:103–117, 1992.
17. D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem.* PWS Publishing Company, 1997.
18. D. S. Hochbaum. A framework of half integrality and good approximations. Manuscript, UC berkeley, 1997.
19. K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *40th IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.
20. H. W. Kuhn. The Hungarian method of solving the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
21. D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995.