A Tale of Two Methods

Reuven Bar-Yehuda[1] and Dror Rawitz[2]

[1] Department of Computer Science, Technion, Haifa 32000, Israel
reuven@cs.technion.ac.il
[2] Caesarea Rothschild Institute, University of Haifa, Haifa 31905, Israel
rawitz@cri.haifa.ac.il

Abstract. We describe two widely used methods for the design and analysis of approximation algorithms, the *primal-dual schema* and the *local ratio technique*. We focus on the creation of both methods by revisiting two results by Bar-Yehuda and Even—the linear time primal-dual approximation algorithm for set cover, and its local ratio interpretation. We also follow the evolution of the two methods by discussing more recent studies.

1 Introduction

We describe two approximation methods for solving combinatorial optimization problems, the *primal-dual schema* and the *local ratio technique*. We specifically focus on the contribution of two papers written by Reuven Bar-Yehuda and Shimon Even in the early 1980's. In their first paper [8] Bar-Yehuda and Even presented a linear programming (LP) based approximation algorithm for the *set cover problem*, and for its the well known special case, the *vertex cover problem*. The idea of using linear programming for approximating set cover was not new—it was used before by Chvátal [17] and Hochbaum [24]. However, the specific way in which linear programming was used was new. Bar-Yehuda and Even's algorithm [8] constructs simultaneously a primal integral solution and a dual feasible solution without solving either the primal or dual programs. Their algorithm was the first to operate in this way, which later became known as the primal-dual schema. The local ratio technique was first used about a couple of years later in a second paper by Bar-Yahuda and Even [9] that deals with the set cover problem. In this paper they presented a local ratio analysis of the algorithm from [8]. They also developed a $(2 - \frac{\log \log n}{2 \log n})$-approximation algorithm for the vertex cover problem, which is partially based on the local-ratio technique. Over the years the two methods have become immensely popular. Numerous algorithms which use either the primal-dual schema or the local ratio technique were published. Almost two decades later, Bar-Yehuda and Rawitz [13] proved that the two methods are actually equivalent.

Before going any further, we present some basic concepts relating to *combinatorial optimization* and *approximation algorithms*. An optimization problem is a problem consisting of a set of possible instances. Each possible instance has a set of candidate solutions, called *feasible solutions*, each of which is associated with a weight. In a minimization (resp., maximization) problem our goal is to find a feasible solution of minimum (resp., maximum) weight. Such a solution is called an *optimal solution*. The weight of an optimal solution is call the *optimum*. For example, in the *vertex cover problem*, an instance consists of a simple graph $G = (V, E)$, and a weight function w on the vertices. A solution is a set of vertices, and a feasible solution is a subset $U \subseteq V$ such that each edge in E has at least one end-point in U. Such a feasible solution is called a *vertex cover*. The weight of a vertex cover U is the total weight of the vertices in U. In the vertex cover problem our goal is to obtain a minimum weight vertex cover. The special case in which $w(u) = 1$ for every $u \in V$ is referred to as the *unweighted vertex cover problem*. In this problem, our goal is to find a vertex cover of minimum cardinality.

Since the vertex cover problem and many other optimization problems are NP-hard, we are forced to compromise. Instead of seeking algorithms that compute optimal solutions in polynomial time, we are willing to settle for efficient algorithm that compute near optimal solutions, or *approximate solutions*. A solution whose weight is within a factor of r of the optimum is called *r-approximate*. An *r-approximation algorithm* is an algorithm that computes r-approximate solutions.

For example, consider Algorithm **UnweightedVC** which is a 2-approximation algorithm for unweighted vertex cover due to Gavril (see [20]).

Algorithm 1 - UnweightedVC(G): a 2-approximation algorithm for vertex cover

---

1: $U \leftarrow \emptyset$
2: **while** there exists an uncovered edge **do**
3:    Let (u, v) be an uncovered edge
4:    $U \leftarrow U \cup \{u, v\}$
5: **end while**
6: Return U

Clearly, this algorithm runs in linear time. Also, U is a vertex cover because this is the termination condition of the algorithm. However, how close is the size of the solution U to the size of an optimal vertex cover? We show that the size of U is quite close to the optimum by proving that it is not more than twice the optimum. Denote by M the set of edges that are considered in Line 3. Clearly, M is a *maximal matching*. Since there are no two edges in M that share a common vertex, any vertex cover must be at least as large as M. Hence, if U^* is an optimal vertex cover, then $|U| = 2|M| \leq 2|U^*|$.

Consider the analysis of Algorithm **UnweightedVC**. It is based on the following simple idea. First, we find a lower bound on the optimum value and then we show that the size of the solution computed by the algorithm is bounded by r times the lower bound, where r is the approximation ratio of the algorithm. The bound in our case is the size of M. This theme is widely used in the field of approximation algorithms, especially in approximation algorithm that are based on linear programming—many combinatorial optimization problems can be expressed as linear integer programs, and the value of an optimal solution to their *LP-relaxation* provides the desired bound.

As we shall see in the sequel, algorithms that fall within the scope of either the primal-dual schema or the local ratio technique use a variation on the lower bound idea (or, upper bound, in the maximization case). Let W denote the weight of the solution computed by the algorithm. Instead of finding directly some lower bound B on the optimum such that $W \leq r \cdot B$, we break down the weight of the solution into a sum of partial weights $W = W_1 + \ldots + W_k$. Then, for each such partial weight W_i we find a "partial" lower bound B_i such that $W_i \leq r \cdot B_i$. Our solution is r-approximate since the sum of the partial lower bounds is not greater than the optimum. In both methods the breakdown of W is determined by the manner in which the solution is constructed by the algorithm. In fact, the algorithm constructs the solution in such a manner as to ensure that such a breakdown exists. The breakdown is done in steps, where in the ith step, the algorithm determines the ith partial weight, and the ith lower bound B_i. In the primal-dual schema the partial weight and bound are induced by an increase of the dual solution, while in the local ratio technique they are determined by the construction of a weight function.

The remainder of this essay is organized as follows. In Section 2 we present several basic results in the area of linear programming, and formally define the problems that we consider in this essay. Bar-Yehuda and Even's [8] primal-dual approximation algorithm for set cover is presented, in *hitting set terms*, in Section 3. Afterwards, we give a general description of the schema, and demonstrate it on an extension of the hitting set problem called *generalized hitting set*. The local ratio version of the approximation algorithm for set cover [9] is given in Section 4. This section also contains a general description of the local ratio technique, and a local ratio algorithm for generalized hitting set. Finally, in Section 5 we survey results that were obtained in both methods during the last two decades, and discuss the connection between the two methods.

2 Preliminaries

2.1 Linear Programming

In this section we state several basic facts from the theory of linear programming. Note that the section is written in terms of minimization problems. (Similar arguments can be made in the maximization case.) For more details about linear programming the reader is referred to, e.g., [28, 29, 31].

Consider the following linear integer program:

$$\begin{aligned} \min \sum_{j=1}^{n} c_j x_j & & \\ \text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \geq b_i & \qquad \forall i \in \{1, \ldots, m\} \\ x_j \in \mathbb{N} & \qquad \forall j \in \{1, \ldots, n\} \end{aligned} \tag{IP}$$

The LP-relaxation of IP is obtained by removing the integrality constraints:

$$\begin{aligned} \min \sum_{j=1}^{n} c_j x_j & & \\ \text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \geq b_i & \qquad \forall i \in \{1, \ldots, m\} \\ x_j \geq 0 & \qquad \forall j \in \{1, \ldots, n\} \end{aligned} \tag{P}$$

Let $\textsc{opt}(\text{IP})$ and $\textsc{opt}(\text{P})$ denote the optimum of IP and P, respectively. Notice that any feasible solution of IP is also feasible with respect to P. Hence,

Observation 1 $\textsc{opt}(P) \leq \textsc{opt}(IP)$.

We refer to P as the *primal* linear program. The following linear program is the *dual* of P:

$$\begin{aligned} \max \sum_{i=1}^{m} b_i y_i & & \\ \text{s.t.} \sum_{i=1}^{m} a_{ij} y_i \leq c_j & \qquad \forall j \in \{1, \ldots, n\} \\ y_i \geq 0 & \qquad \forall i \in \{1, \ldots, m\} \end{aligned} \tag{D}$$

A solution of P is called a *primal solution*, and a solution of D is called a *dual solution*. A solution of IP is referred to as an *integral primal solution*.

The connection between the primal and dual optima is given by the following two theorems (the second is given without proof):

Theorem 2 (Weak Duality). *Let x and y be a pair of primal and dual solutions. Then, $b^T y \leq c^T x$.*

Proof.

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \geq \sum_{i=1}^{m} b_i y_i \tag{1}$$

where the first inequality follows from a summation of the dual constraints, and the second follows from a summation of the primal constraints. □

Theorem 3 (Strong Duality). *Let x^* and y^* be a pair of optimal primal and dual solutions. Then, $b^T y^* = c^T x^*$.*

It follows that

Observation 4 *Let x be an integral primal solution, and let y be a dual solution. Then, $b^T y \leq \text{OPT}(D) = \text{OPT}(P) \leq \text{OPT}(IP) \leq c^T x$.*

The Strong Duality Theorem provides us with a way to characterize a primal-dual pair of optimal solutions.

Theorem 5 (Complementary Slackness Conditions). *Let x and y be a pair of primal and dual solutions. Then, x and y are optimal if and only if the following conditions, called the* complementary slackness conditions*, are satisfied:*

$$\begin{aligned}
&\textit{Primal conditions: } \forall j,\ x_j > 0 \Rightarrow \sum_{i=1}^{m} a_{ij} y_i = c_j \\
&\textit{Dual conditions: } \quad \forall i,\ y_i > 0 \Rightarrow \sum_{j=1}^{n} a_{ij} x_j = b_i
\end{aligned}$$

Proof. First, assume x and y are optimal. By the Strong Duality Theorem it follows that $c^T x = b^T y$. Therefore, the inequalities in Equation (1) become equalities:

$$\sum_{j=1}^{n} c_j x_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i = \sum_{i=1}^{m} b_i y_i \tag{2}$$

The primal complementary slackness conditions are implied by the first equality, and the dual conditions are implied by the third equality.

For the other direction assume that the complementary slackness conditions are satisfied. In this case Equality (2) is satisfied as well, and therefore $c^T x = b^T y$. x and y are optimal by the Weak Duality Theorem. □

2.2 The Problems

Recall that, in the *vertex cover problem*, an instance consists of a simple graph $G = (V, E)$, and a weight function w on the vertices, and our goal is to obtain a minimum weight vertex cover. Hence, the vertex cover problem can be formulated by the following linear integer program:

$$\begin{array}{lll}
\min & \sum_{u \in V} w(u) x_u & \\
\text{s.t.} & x_u + x_v \geq 1 & \forall (u,v) \in E \\
& x_u \in \{0,1\} & \forall u \in V
\end{array} \tag{VC}$$

where $x_u = 1$ if and only if u is in the vertex cover. The LP-relaxation of VC is obtained by replacing the integrality constraints by: $x_u \geq 0$ for every $u \in V$.

(Notice that the possible inequalities of the form $x_u \leq 1$ are redundant.) We denote the LP relaxation of VC by VC-P. (Henceforth, a -P suffix denotes the LP-relaxation of a linear integer program, while a -D suffix denotes the dual of the LP-relaxation.)

The *hitting set problem* is defined as follows. The input consists of a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of the ground set U of size n. Each element $u \in U$ is associated with a positive weight $w(u)$. A set H is said to *hit* a subset S if $H \cap S \neq \emptyset$. A *hitting set* is a set $H \subseteq U$ that hits every subset $S \in \mathcal{S}$. In the hitting set problem our goal is find a hitting set of minimum total weight. Given a hitting set instance, we denote by $\mathcal{S}(u)$ the collection of sets that contain u, i.e., $\mathcal{S}(u) \triangleq \{S : u \in S\}$. We define $s_{\max} \triangleq \max_{S \in \mathcal{S}} |S|$. Note that the vertex cover problem is a special case of hitting set in which all sets are of size two, and hence $s_{\max} = 2$ in this special case. We also note that some of the results in this survey were originally written in terms of the *set cover problem.* In the *set cover* problem we are given a collection of sets $\mathcal{S}$ of the ground set U, and a weight function on the subsets. The objective is to find a minimum weight collection of sets that covers all elements, or a minimum weight *set cover.* It is easy to see that set cover and hitting set are equivalent problems in the sense that each is obtained from the other by switching the roles of sets and elements.

The hitting set problem can be formulated by the following linear integer program:

$$\begin{aligned} \min \ & \sum_{u \in U} w(u) x_u \\ \text{s.t.} \ & \sum_{u \in S} x_u \geq 1 && \forall S \in \mathcal{S} \\ & x_u \in \{0,1\} && \forall u \in U \end{aligned} \tag{HS}$$

where $x_u = 1$ if and only if u is in the hitting set. The LP-relaxation of HS is obtained by replacing the integrality constraints by: $x_u \geq 0$ for every $u \in U$. We denote that LP relaxation by HS-P. The dual of HS-P is:

$$\begin{aligned} \max \ & \sum_{S \in \mathcal{S}} y_S \\ \text{s.t.} \ & \sum_{S \in \mathcal{S}(u)} y_S \leq w(u) && \forall u \in U \\ & y_S \geq 0 && \forall S \in \mathcal{S} \end{aligned} \tag{HS-D}$$

In the *generalized hitting set problem* we are also given of a collection of subsets $\mathcal{S}$ of the ground set U, and our goal is to hit the sets in $\mathcal{S}$ by using elements from U. However, in this case, we are allowed not to hit a set S, provided that we pay a tax $w(S)$. Hence, the weight function w is define on both the elements and the subsets. Formally, the input is a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of the ground set $U = \{1, \ldots, n\}$, and a weight function on the elements and subsets, and our goal is to find a minimum-weight set $H \subseteq U$, where the weight of H is the weight of the elements in H and the weight of the sets that are not hit by H. The hitting set problem is the special case where the tax $w(S)$ is infinite for every set $S \in \mathcal{S}$.

In order to formulate generalized hitting set using a linear program it would be convenient to slightly change the problem definition. Instead of simply searching for a set of elements H, we shall search for a set of elements H and a sub-collection of sets $\mathcal{T}$ such that for every set S either S is hit by H, or it is contained in $\mathcal{T}$, i.e., we seek a pair $(H, \mathcal{T})$ where $H \subseteq U$, $\mathcal{T} \subseteq \mathcal{S}$, and for all $S \in \mathcal{S}$, either $H \cap S \neq \emptyset$ or $S \in \mathcal{T}$. This means that we allow a set S to be both hit by H and contained in $\mathcal{T}$. Clearly, for a given generalized hitting set instance, the optima of both problems are the same. Moreover, any solution for the second definition can be easily turned into a solution for the first. Hence, the generalized hitting set problem can be formalized as follows:

$$
\begin{array}{lll}
\min & \sum_{u \in U} w(u) x_u + \sum_{S \in \mathcal{S}} w(S) x_S & \\
\text{s.t.} & \sum_{u \in S} x_u + x_S \geq 1 & \forall S \in S \\
& x_u \in \{0,1\} & \forall u \in U \\
& x_S \in \{0,1\} & \forall S \in \mathcal{S}
\end{array}
\tag{GHS}
$$

where $x_u = 1$ if and only if the element u is contained in H, and $x_S = 1$ if and only if the subset S is contained in $\mathcal{T}$. As usual, the LP-relaxation is obtained by replacing the integrality constraints by: $x_u \geq 0$ for every $u \in U$ and $x_S \geq 0$ for every $S \in \mathcal{S}$. We denote that LP relaxation by GHS-P. The dual is:

$$
\begin{array}{lll}
\max & \sum_{S \in \mathcal{S}} y_S & \\
\text{s.t.} & \sum_{S \in \mathcal{S}(u)} y_S \leq w(u) & \forall u \in U \\
& y_S \leq w(S) & \forall S \in \mathcal{S} \\
& y_S \geq 0 & \forall S \in \mathcal{S}
\end{array}
\tag{GHS-D}
$$

Notice that a generalized hitting set instance can be viewed as a hitting set instance in which each set S contains a unique element u_S whose weight is $w(u_S) \triangleq w(S)$. This way, we can pay $w(S)$ for the element u_S instead of paying the tax $w(S)$ for not hitting S. In the sequel we present several $s_{\max}$-approximation algorithms for hitting set. It follows that these algorithm can be used to obtain $(s_{\max} + 1)$-approximate solutions for generalized hitting set. However, we also show how to obtain $s_{\max}$-approximate solutions for generalized hitting set.

An important notion in the design of approximation algorithms using primal-dual or local ratio is the notion of *minimal solutions*. A feasible solution is said to be minimal with respect to set inclusion (or minimal for short) if all its proper subsets are not feasible. Minimal solutions arise naturally in the context of *covering problems*, which are the problems for which feasible solutions have the property of being monotone inclusion-wise, that is, the property that adding items to a feasible solution cannot render it infeasible. For example, adding an element to a hitting set yields a hitting set, so *hitting set* is a covering problem. In contrast, adding an edge to a spanning tree does not yield a tree, so *minimum spanning tree* is not a covering problem. However, if instead of focusing only on trees, we consider all sets of edges that intersect all non trivial cuts in the given graph the problem becomes a covering problem.

It is easy to see that generalized hitting set (under the second definition) is a covering problem. The following observation formalizes the fact that it makes no sense to add a set S to $\mathcal{T}$ if $H \cap S \neq \emptyset$.

Observation 6 *Let $(H, \mathcal{T})$ be a minimal solution, and let x be the incidence vector of $(H, \mathcal{T})$. Then, (i) $x_S = 1$ if and only if $\sum_{u \in S} x_u = 0$, and (ii) $\sum_{u \in S} x_u + x_S \leq s_{\max}$.*

We note that the use of minimality in the context of the generalized hitting set problem is somewhat artificial. However, it will assist us in demonstrating the use of minimality in the design of primal-dual and local ratio approximation algorithms for covering problems.

3 The Primal-Dual Schema

In this section we present the primal-dual $s_{\max}$-approximation algorithm for hitting set from [8]. Afterwards we give a general description of the primal-dual schema, and demonstrate it on the generalized hitting set problem.

An r-approximation algorithm for a minimization problem that is based on a primal-dual analysis produces an integral primal solution x and a dual solution y such that the weight of the primal solution is not more than r times the value of dual solution. Namely, it produces an integral primal solution x and a dual solution y such that

$$c^T x \leq r \cdot b^T y \tag{3}$$

The integral primal solution x is r-approximate due to Observation 4.

There are several ways to find such a pair of primal and dual solutions. The first one to do so was Chvátal [17], who proved that the greedy algorithm for hitting set computes $\mathcal{H}_m$-approximate solutions, where $\mathcal{H}_m$ is the mth harmonic number. (Recall that, in hitting set terms, m is the number of sets.) In his analysis he obtained an *infeasible* dual solution whose value as not less than the weight of the integral primal solution that was computed by the algorithm. Then, he showed that if the dual solution is divided by $\mathcal{H}_m$ it becomes feasible. This method was later called *dual fitting*, and the feasible solution was referred to as *shrunk dual*. (See [32, 26] for more details.)

Hochbaum [24] presented several $s_{\max}$-approximation algorithms for hitting set that require the solution of a linear program. The first algorithm is as follows: (i) compute an optimal solution y^* of HS-D, and (ii) return $H_D = \{u : \sum_{S \in \mathcal{S}(u)} y^*_S = w(u)\}$. We show that H_D is a hitting set. Assume by contraposition that H_D is not a hitting set. Then, there exists a set S such that $S \cap H_D = \emptyset$. Let $\varepsilon = \min_{u \in S}\{w(u) - \sum_{S \in \mathcal{S}(u)} y^*_S\}$. Clearly, $\varepsilon > 0$. Furthermore, if we raise y^*_S by ε it remains feasible in contradiction to the optimality y^*. Next, we show that H_D is $s_{\max}$-approximate. Let x_D be the incidence vector of H_D. Then,

$$\begin{aligned}
w(H_D) &= \sum_{u \in U} w(u) x_u \\
&= \sum_{u \in U} x_u \sum_{S \in \mathcal{S}(u)} y^*_S \\
&= \sum_{S \in \mathcal{S}} y^*_S \sum_{u \in S} x_u \\
&\leq s_{\max} \sum_{S \in \mathcal{S}} y^*_S \\
&\leq s_{\max} \cdot \mathrm{OPT}
\end{aligned}$$

and we are done.

An $s_{\max}$-approximate hitting set can also be found by solving the primal linear program HS-P. Consider the following algorithm: (i) compute an optimal solution x^* of HS-P, and (ii) return $H_P = \{u : x^*_u > 0\}$. H_P must be a hitting set, since otherwise, x^* is not feasible. Moreover, by the complementary slackness conditions $H_P \subseteq H_D$ (where H_D is defined as above). Hence, H_P is $s_{\max}$-approximate as well. Note that it is even enough to consider only elements whose primal variable is at least $1/s_{\max}$. That is, the set $H'_P = \{u : x^*_u \geq 1/s_{\max}\}$ is also an $s_{\max}$-approximate solution. H'_P is feasible since there exists $u \in S$ such that $x^*_u \geq 1/s_{\max}$ for every subset $S \in \mathcal{S}$, and H'_P is $s_{\max}$-approximate since $H'_P \subseteq H_P$.

Following the work of Hochbaum [24], Bar-Yehuda and Even [8] presented another $s_{\max}$-approximation algorithm for hitting set that uses primal-dual arguments. As opposed to Hochbaum's algorithm, this algorithm is not based on finding an *optimal* dual (or primal) solution, and therefore it is more efficient. The key observation that was made by Bar-Yehuda and Even [8] is that the dual solution, y^*, used in Hochbaum's analysis does not have to be optimal. A dual solution y is called *maximal* if an increase in y_i makes y infeasible, for any i. Clearly, an optimal dual solution is also maximal. It is not hard to verify that $H'_D = \{u : \sum_{S \in \mathcal{S}(u)} y_S = w(u)\}$ is a hitting set for any maximal (and not necessarily optimal) dual solution y. Hence, Hochbaum's analysis stays intact when a maximal dual solution is used (and H_D is replaced by H'_D). The improved running time is due to the fact that a simple greedy algorithm can compute a maximal dual solution in linear time. Algorithm **PD-HS** is the algorithm from [8] given in terms of hitting set.

It is not hard to verify that the running time of Algorithm **PD-HS** is $O(\sum_{S \in \mathcal{S}} |S|)$, which means that it runs in linear time. Observe that, in every iteration, y_S is raised as much as possible while maintaining feasibility, hence y is a maximal dual solution. Hence, the set of elements whose corresponding dual constraint is tight (i.e., H'_D) constitute an $s_{\max}$-approximate hitting set. Algorithm **PD-HS** does not return the set of elements whose corresponding

Algorithm 2 - PD-HS$(U, \mathcal{S}, w)$: a primal-dual $s_{\max}$-approximation algorithm for hitting set

1: $H \leftarrow \emptyset$
2: $y \leftarrow 0$
3: **while** $\mathcal{S} \neq \emptyset$ **do**
4: Let $S \in \mathcal{S}$
5: $v \leftarrow \text{argmin}_{u \in S} \left\{ w(u) - \sum_{S' \in \mathcal{S}(u)} y_{S'} \right\}$
6: $y_S \leftarrow w(v) - \sum_{S' \in \mathcal{S}(v)} y_{S'}$
7: $H \leftarrow H \cup \{v\}$
8: $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{S}(v)$
9: **end while**
10: Return H

dual constraint is tight. However, an element v may be added to H only if its corresponding dual constraint is tight (i.e., $H \subseteq H'_D$). Moreover, every subset S contains at least one such element. Hence, H is feasible and therefore also $s_{\max}$-approximate.

It is important to notice that since the choice of v (in Line 6) is made according to the tightness of the dual constraints, and not according to the values of the dual variables, it is enough to compute, in every iteration, the tightness of the dual constraints, instead of maintaining the dual solution y. The actual values of the dual variables are needed only for purposes of analysis. Hence, Lines 5–6 of Algorithm **PD-HS** can be replaced with the following two lines:

5: $v \leftarrow \text{argmin}_{u \in S} \{w(u)\}$
6: For every $u \in S$ do: $w(u) \leftarrow w(u) - w(v)$

In fact, the original algorithm was presented in this way in [8].

Algorithm **PD-HS** computes an integral primal solution x, the incidence vector of H, and a dual solution y such that the weight of x is bounded by $s_{\max}$ times the value of y. This seems like a neat trick, but can we use this idea in order to approximate other problems? We shall see that the connection between x and y is somewhat more complicated than what is implied by the analysis of Algorithm **PD-HS**. Clearly, Algorithm **PD-HS** picks only elements whose corresponding dual constraint is tight. Hence, if $x_u = 1$ (i.e., $u \in H$) then $\sum_{S \in \mathcal{S}(u)} y_S = w(u)$. Now, consider a set $S \in \mathcal{S}$, and the corresponding constraint $\sum_{u \in S} x_u \geq 1$. Clearly, $\sum_{u \in S} x_u \leq s_{\max}$ for any S such that $y_S > 0$ (or, for any other S). Putting it all together we get that x and y satisfy the following conditions:

$$\forall u \in U,\ x_u > 0 \Rightarrow \sum_{S \in \mathcal{S}(u)} y_S = w(u)$$

$$\forall S \in \mathcal{S},\ y_S > 0 \Rightarrow 1 \leq \sum_{u \in S} x_u \leq s_{\max}$$

The first set of conditions are exactly the primal complementary slackness conditions, while the second is a relaxation of the dual conditions. Moreover, the relaxation factor is exactly the approximation ratio of Algorithm **PD-HS**. As we shall see this idea is not limited to the hitting set problem.

Let x be an integral primal solution, and let y be a dual solution. Also, assume that x and y satisfy the following *relaxed complementary slackness conditions*:

$$\begin{array}{ll} \text{Primal conditions:} & \forall j,\ x_j > 0 \Rightarrow \sum_{i=1}^{m} a_{ij} y_i = c_j \\ \text{Relaxed dual conditions:} & \forall i,\ y_i > 0\ \Rightarrow b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq r \cdot b_i \end{array}$$

Then,

$$\sum_{j=1}^{n} c_j x_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq r \cdot \sum_{i=1}^{m} b_i y_i$$

which means that x is r-approximate. Hence, we have found a way to compute a pair of integral primal and dual solutions that satisfy Inequality (3).

Indeed, a typical primal-dual algorithm computes a primal-dual pair (x, y) that satisfies the relaxed complementary slackness conditions. Moreover, a primal-dual algorithm usually constructs the primal-dual pair in such a way that the relaxed complementary slackness conditions are satisfied throughout its execution. It starts with an infeasible primal solution and a feasible dual solution (usually, $x = 0$ and $y = 0$). It iteratively raises the dual solution, and improves the feasibility of the primal solution while maintaining the following two invariants: (i) a primal variable is increased only if its corresponding primal condition is satisfied, and (ii) a dual variable is increased only if its corresponding relaxed dual condition is satisfied. (We note that many primal-dual algorithms change several dual variables in each iteration. However, it can be shown that it is enough to raise only a single dual variable in each iteration [15, 13].) Hence, an iteration of a primal-dual r-approximation algorithm (for a covering problem) can be informally described as follows:

1. Find a primal constraint, $\alpha x \geq \beta$, such that $\alpha x \leq r \cdot \beta$ for every feasible solution x.
2. Raise the dual variable that corresponds to the above primal constraint until some dual constraint becomes tight.
3. Add an element whose corresponding dual constraint is tight to the primal solution.

Steps (1) and (2) ensure that the relaxed dual conditions are satisfied, while Step (3) ensures that the primal conditions are satisfied. The reader is referred to [15, 13] for a formal description.

In many primal-dual algorithms the Step (1) is slightly modified. Instead of finding a primal constraint for which $\alpha x \leq r \cdot \beta$ for every feasible solution x, it is enough to find a primal constraint $\alpha x \geq \beta$, for which $\alpha x \leq r \cdot \beta$ for every minimal solution x (or, sometimes, for every feasible solution x that satisfies some other property $\mathcal{P}$). Such a constraint is called *r-effective* (or *r-effective with respect to* $\mathcal{P}$). In this case, the algorithm must compute a minimal solution x (or a solution x that satisfies $\mathcal{P}$), since otherwise the dual solution y do not satisfy the relaxed dual complementary slackness conditions at termination. Primal-dual algorithms that compute minimal solutions usually use a primal pruning procedure that is sometimes referred to as *reverse deletion*. Algorithms that use a property $\mathcal{P}$ other than minimality use some sort of solution correction procedure that depend on $\mathcal{P}$. (See [15, 13] for more details.)

We demonstrate the above ideas on the generalized hitting set problem. Algorithm **PD-GHS** is an $s_{\max}$-approximation algorithm for generalized hitting set that was presented by Bar-Yehuda and Rawitz [13].

Algorithm 3 - PD-GHS$(U, \mathcal{S}, w)$: a primal-dual $s_{\max}$-approximation algorithm for generalized hitting set

1: $H \leftarrow \emptyset, \mathcal{T} \leftarrow \emptyset$
2: $y \leftarrow 0$
3: **for all** $S \in \mathcal{S}$ **do**
4:    Raise y_S until some dual constraint becomes tight
5:    **if** there exists an element $u \in S$ whose dual constraint became tight **then**
6:       $H \leftarrow H \cup \{u\}$
7:    **else**
8:       $\mathcal{T} \leftarrow \mathcal{T} \cup \{S\}$
9:    **end if**
10: **end for**
11: $\mathcal{T} \leftarrow \mathcal{T} \setminus \bigcup_{u \in H} \mathcal{S}(u)$.
12: Return $(H, \mathcal{T})$

We show that Algorithm **PD-GHS** computes a pair of minimal primal solution and dual solution that satisfies the relaxed complementary slackness conditions. First, $(H, \mathcal{T})$ is minimal due to Line 11. Also, the primal conditions are satisfied by the construction of the primal solution. Now, consider a set $S \in \mathcal{S}$ and its corresponding primal constraint: $\sum_{u \in S} x_u + x_S \geq 1$. According to Observation 6 we know that $\sum_{u \in S} x_u + x_S \leq s_{\max}$ for every minimal solution x. Hence, any minimal solution satisfies the relaxed dual slackness condition corresponding to S. It follows that Algorithm **PD-GHS** computes $s_{\max}$-approximate solutions.

We note that in some cases proving that the relaxed dual slackness conditions are satisfied for some r may be a difficult task, e.g., the 2-approximation algorithms for the *feedback vertex set problem* [16]).

We also remark that most primal-dual algorithms in the literature are based on a predetermined LP formulation, and therefore several dual variable are changed in each iteration of the algorithm. Hence, most primal-dual algorithms do not refer explicitly to the relaxed complementary slackness conditions. As mentioned above, such algorithms can be altered such that only a single dual variable is changed in each iteration (see [15, 13]). After doing so these analyses can be easily explained using the relaxed conditions. It is important to note that the combinatorial properties of the problem that were used in the analysis are usually presented in a much clearer fashion when the analysis change only a single dual variable in each iteration and is based on the relaxed complementary slackness conditions. Hence, such analyses tend to be simpler and more elegant.

4 Local Ratio

We re-consider Gavril's 2-approximation algorithm for unweighted vertex cover (Algorithm **UnweightedVC**). In each iteration the algorithm picks two vertices u and v that cover the uncovered edge (u, v). Since this edge must be covered, any vertex cover must contain at least one of the vertices. Hence, if we take both u and v we decrease the optimum by at least one, while adding not more than two vertices to the solution. Notice that this argument is local in the sense that it refers separately to any edge in M. (Recall that M is the maximal matching constructed by the algorithm.) This simple idea is at the heart of the local ratio technique.

We show how to extend this algorithm to an $s_{\max}$-approximation algorithm for the weighted hitting set problem. Imagine that we have to actually purchase the elements we select as our solution. Rather than somehow deciding on which elements to buy and then paying for them, we adopt the following strategy. We repeatedly select an element and pay for it. However, the amount we pay need not cover its entire cost; we may return to the same vertex later and pay some more. In order to keep track of the payments, whenever we pay ε for a vertex, we lower its marked price by ε. When the marked price of an element drops to zero, we are free to take it, as it has been fully paid for. The heart of the matter is the rule by which we select the element and decide on the amount to pay for it. Actually, we select up to $s_{\max}$ elements each time and pay ε for each, in the following manner. We select any subset S whose elements have non-zero weight, and pay $\varepsilon = \min_{u \in S} w(u)$ for every element in S. As a result, the weight of at least one of the elements drops to zero. After $O(n)$ rounds, prices drop sufficiently so that every set contains an element of zero weight. Hence, the set of all zero-weight elements is a hitting set.

We formalize the above discussion by Algorithm **LR-HS** which is a linear time $s_{\max}$-approximation algorithm that was presented by Bar-Yehuda and Even [9]. (The original algorithm was presented in set cover terms.) We say that a set is *positive* if all its elements have strictly positive weights. Notice that on instances of unweighted vertex cover it is identical to Gavril's 2-approximation algorithm.

To formally analyze the algorithm consider the ith iteration. Let S_i be the set that was selected in this iteration, and let ε_i be the weight of the minimum weight element in S_i. Since every hitting set must contain at least one element in S_i, decreasing the weight of the elements in S_i by ε_i lowers the weight of every hitting set by at least ε_i. Hence, the optimum must also decrease by at

Algorithm 4 - LR-HS$(U, \mathcal{S}, w)$: a local ratio $s_{\max}$-approximation algorithm for hitting set

1: **while** there exists a positive set S **do**
2: $\quad \varepsilon \leftarrow \min_{u \in S} \{w(u)\}$
3: $\quad$ For every $u \in S$ do: $w(u) \leftarrow w(u) - \varepsilon$
4: **end while**
5: Return the set $H = \{u : w(u) = 0\}$

least ε_i. Thus, in the ith round we pay $s_{\max} \cdot \varepsilon_i$ and lower the optimum by at least ε_i. Since H is a zero weight hitting set (with respect to the final weights), the optimum has decreased to zero. Hence, $\text{OPT} \geq \sum_i \varepsilon_i$. One the other hand, since our payments fully cover H, its weight is bounded by $\sum_i s_{\max} \cdot \varepsilon_i$. H is $s_{\max}$-approximate, because $w(H) \leq \sum_i s_{\max} \cdot \varepsilon_i \leq s_{\max} \cdot \text{OPT}$.

It is interesting to note that the proof that the solution found is $s_{\max}$-approximate does not depend on the actual value of ε in any given iteration. In fact, any value between 0 and $\min_{u \in S} \{w(u)\}$ would yield the same result (by the same arguments). We chose $\min_{u \in S} \{w(u)\}$ for the sake of efficiency. This choice ensures that the number of elements with positive weight strictly decreases with each iteration.

In Algorithm **LR-HS** we have paid $s_{\max} \cdot \varepsilon$ for lowering OPT by at least ε in each round. Other local ratio algorithms can be explained similarly—one pays in each round at most $r \cdot \varepsilon$, for some r, while lowering OPT by at least ε. If the same r is used in all rounds, the solution computed by the algorithm is r-approximate. This idea works well for several problems. However, it is not hard to see that this idea works mainly because we make a down payment on several elements, and we are able to argue that OPT must drop by a proportional amount because every solution must contain one of these elements. This localization of the payments is at the root of the simplicity and elegance of the analysis, but it is also the source of its weakness: how can we design algorithms for problems in which no single element (or set of elements) is necessarily involved in every optimal solution? For example, consider the *feedback vertex set* problem, in which we are given a graph and a weight function on the vertices, and our goal is to remove a minimum weight set of vertices such that the remaining graph contains no cycles. Clearly, it is not always possible to find a constant number of vertices such that at least one of them is part of every optimal solution!

It helps to view the payments made by the algorithm as the subtraction of a new weight function w_1 from the current weight function w. For example, examine an iteration of Algorithm **LR-HS**. The action taken in Line 3 is equivalent to defining a new weight function:

$$w_1(u) = \begin{cases} \varepsilon & u \in S, \\ 0 & u \notin S, \end{cases}$$

and subtracting it from w. The analysis above implies that:

Observation 7 *Every hitting set is $s_{\max}$-approximate with respect to w_1.*

Hence, any $s_{\max}$-approximate hitting set H with respect to $w - w_1$ is also $s_{\max}$-approximate with respect to w_1. By the following theorem H is also $s_{\max}$-approximate with respect to w.

Theorem 8 (Local Ratio Theorem [9, 6]). *Let $(\mathcal{F}, w)$ be a minimization problem, where $\mathcal{F}$ is a set of constraints on $x \in \mathbb{R}^n$, and $w \in \mathbb{R}^n$ is a weight function. Also, let w_1 and w_2 be weight functions such that $w = w_1 + w_2$. Then, if x is r-approximate with respect to $(\mathcal{F}, w_1)$ and with respect to $(\mathcal{F}, w_2)$, then x is r-approximate with respect to $(\mathcal{F}, w)$.*

Proof. Let x^*, x_1^*, x_2^* be optimal solutions with respect to $(\mathcal{F}, w), (\mathcal{F}, w_1)$, and $(\mathcal{F}, w_2)$, respectively. Then, $wx = w_1 x + w_2 x \leq r \cdot w_1 x_1^* + r \cdot w_2 x_2^* \leq r \cdot (w_1 x^* + w_2 x^*) = r \cdot wx^*$. □

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

This idea of weight decomposition leads us to the to Algorithm **Recursive-LR-HS** which is a recursive version of Algorithm **LR-HS**.

Algorithm 5 - Recursive-LR-HS$(U, \mathcal{S}, w)$: a local ratio $s_{\max}$-approximation algorithm for hitting set

if $\mathcal{S} = \emptyset$ **then**
  Return $\emptyset$
end if
Let $S \in \mathcal{S}$
$v \leftarrow \text{argmin}_{u \in S} \{w(u)\}$
$\varepsilon \leftarrow w(v)$
Define $w_1(u) = \begin{cases} \varepsilon & u \in S, \\ 0 & u \notin S, \end{cases}$
$H \leftarrow \{v\} \cup$ **Recursive-LR-HS**$(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v), w - w_1)$
Return H

We first note that this algorithm is slightly different from Algorithm **LR-HS**, since not all the vertices that have zero weight at the recursive base are necessarily taken into the solution. (A similar pruning procedure can be added to Algorithm **LR-HS** as well.)

Since Algorithm **Recursive-LR-HS** is recursive, it is natural to use induction in its analysis. First, it is not hard to show that the solution returned is a hitting set by induction on the number of recursive calls. (Note that this number is bounded by the number of elements.) We prove that the solution is $s_{\max}$-approximate by induction. In the base case, $\emptyset$ is an optimal solution. For the inductive step, let H be the solution returned, and denote $w_2 = w - w_1$. By the induction hypothesis $H \setminus \{v\}$ is $s_{\max}$-approximate with respect to $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v), w_2)$. Since $w_2(v) = 0$, the optima of $(\mathcal{S}, U, w_2)$

and of $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v), w_2)$ are the same. Hence, H is $s_{\max}$-approximate with respect to $(\mathcal{S}, U, w_2)$. Due to Observation 7 any hitting set with respect to the instance $(\mathcal{S}, U)$ is $s_{\max}$-approximate with respect to w_1, therefore H is $s_{\max}$-approximate with respect to $(\mathcal{S}, U, w_1)$. Finally, H is $s_{\max}$-approximate with respect to $(\mathcal{S}, U, w)$ as well due to the Local Ratio Theorem.

Observation 7 states that the weight function w_1 is *well behaved*. That is, from its view point all hitting sets weigh roughly the same (up to a multiplitive factor of $s_{\max}$). We formalize the notion of well behaves weight functions.

Definition 9 *A weight function w is said to be* r-effective *with respect to property $\mathcal{P}$ if there exists a number b such that $b \leq wx \leq r \cdot b$ for all feasible solutions x that satisfy $\mathcal{P}$.*

In Algorithm **Recursive-LR-HS** the property $\mathcal{P}$ uses is simply *feasibility*. However, in many local ratio algorithms the property $\mathcal{P}$ is *minimality*, and in this case w is simply called *r-effective*. When $\mathcal{P}$ is satisfies by every solutions w is sometimes called *fully r-effective*.

It turns out that in many cases it is convenient to use algorithms that are based on weight decomposition. This is especially true when the local ratio advancement step includes more than a constant number of elements, or when w_1 is well behaved for solutions that satisfy a certain property (usually, minimal solutions) and not for every solution.

A typical local-ratio r-approximation algorithm for a covering problem is recursive, and works as follows. Given a problem instance with a weight function w, we find a non-negative weight function $w_1 \leq w$ such that (i) every minimal solution is r-approximate with respect to w_1, and (ii) there exists some index j for which $w(j) = w_1(j)$. We subtract w_1 from w, and remove some zero weight element from the problem instance. Then, we recursively solve the new problem instance. If the solution returned is infeasible the above mentioned element is added to it. This way we make sure that the solution is minimal with respect to the current instance. Since the solution for the current instance is r-approximate with respect to both w_1 and $w - w_1$, it is also r-approximate with respect to w by the Local-Ratio Theorem. The base of the recursion occurs when the problem instance has degenerated into an empty instance.

We demonstrate these ideas by presenting an $s_{\max}$-approximation algorithm for the generalized hitting set problem. The algorithm is taken from [11] and is called Algorithm **LR-GHS**. For purposes of conciseness we represent each set S by an element u_S that is contained in S and whose weight is $w(u_S) \triangleq w(S)$. Recall that, this way, we can pay $w(S)$ for the element u_S instead of paying the tax $w(S)$ for not hitting S.

Note that the main difference between Algorithms **Recursive-LR-HS** and **LR-GHS** is the fact that first simply adds an element to the solution found by the recursive call (Line 8), while the latter adds the element only in case the solution returned by the recursive call is infeasible without it (Lines 8-12). As we shall see this modification makes sure that the solution returned is not only feasible but also minimal.

Algorithm 6 - LR-GHS$(U, \mathcal{S}, w)$: a local ratio $s_{\max}$-approximation algorithm for generalized hitting set

if $\mathcal{S} = \emptyset$ **then**
  Return $\emptyset$
end if
Let $S \in \mathcal{S}$
$v \leftarrow \text{argmin}_{u \in S} \{w(u)\}$
$\varepsilon \leftarrow w(v)$
Define $w_1(u) = \begin{cases} \varepsilon & u \in S, \\ 0 & \text{otherwise.} \end{cases}$
$H' \leftarrow$ **LR-GHS**$(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v), w - w_1)$
$H \leftarrow H'$
if H is not feasible **then**
  $H \leftarrow H \cup \{v\}$
end if
Return H

We show that Algorithm **LR-GHS** computes minimal solutions. The proof is by induction on the recursion. At the recursion basis the solution returned is the empty set, which is both feasible and minimal. For the inductive step, we show that $H \setminus \{u\}$ is not feasible for every $u \in H$. First, if $H = H'$, then H is minimal since H' is minimal with respect to $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v))$ by the inductive hypothesis. Consider the case where $H = H' \cup \{v\}$. If $u \neq v$ and $H \setminus \{u\}$ is feasible, then $H' \setminus \{u\}$ is feasible with respect to $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v))$, and therefore H' is not minimal in contradiction to the inductive hypothesis. Also, observe that v is added to H only if H' is not feasible.

It remains to show that Algorithm **LR-GHS** returns $s_{\max}$-approximate solutions. The proof is by induction on the recursion. In the base case the solution returned is the empty set, which is optimal. For the inductive step, H' is $s_{\max}$-approximate with respect to $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v))$, and w_2 by the inductive hypothesis. Since $w_2(v) = 0$, H is also $s_{\max}$-approximate with respect to $(U \setminus \{v\}, \mathcal{S} \setminus \mathcal{S}(v))$, and w_2. Due to Observation 6, and the fact that H is minimal, it is also $s_{\max}$-approximate with respect to w_1. Thus by the Local Ratio Theorem H is $s_{\max}$-approximate with respect to w as well.

5 The Evolution of Both Methods

In this section we follow the evolution of both primal-dual and local ratio.

5.1 Applications to Various Problems

Covering problems and minimal solutions. During the early 1990's the primal-dual schema was used extensively to design and analyze approximation algorithms for *network design problems*, such as the *Steiner tree problem* (see, e.g., [30, 1, 21]). In fact, this line of research has introduced the idea of using minimal solutions to the primal-dual schema. Subsequently, several primal-dual approximation frameworks were proposed. Goemans and Williamson [22] presented a generic primal-dual approximation algorithm based on the hitting set problem. They showed that it can be used to explain many classical (exact and approximation) algorithms for special cases of the *hitting set* problem, such as *shortest path*, *minimum spanning tree*, and *minimum Steiner forest*. Following [21], Bertsimas and Teo [15] proposed a primal-dual framework for covering problems. As in [22] this framework enforces the primal complementary slackness conditions while relaxing the dual conditions. However, in contrast to previous studies, Bertsimas and Teo [15] express each advancement step as the construction of a single valid inequality, and an increase of the corresponding dual variable (as opposed to an increase of several dual variables).

About ten years after the birth of the local ratio technique [9], Bafna et al. [3] extended the technique in order to construct a 2-approximation algorithm for the *feedback vertex set problem*. Their algorithm was the first local ratio algorithm that used the notion of minimal solutions. Following Bafna et al. [3], Fujito [19] presented a generic local ratio algorithm for a certain family of node deletion problems. Later, Bar-Yehuda [6] presented a local ratio framework for covering problems, which extends the one in [19] and can be used to explain many known optimization and approximation algorithms for covering problems.

Other minimization problems. Both primal-dual and local ratio were also applied to non-covering minimization problems. For example, in [11] the local ratio technique was used in the design of a 2-approximation algorithm for bounded integer programs with two variables per constraint. Recently, Guha et al. [23] presented a primal-dual 2-approximation algorithm to the *capacitated vertex cover problem*.

Jain and Vazirani [27] presented a 3-approximation algorithm for the *metric uncapacitated facility location problem*. Their algorithm was the first primal-dual algorithm that approximated a problem whose LP formulation contains inequalities with negative coefficients. However, this algorithm deviates from the primal-dual schema. Their algorithm does not employ the usual mechanism of relaxing the dual complementary slackness conditions, but rather it relaxes the *primal* conditions. (Note that this algorithm has a non-LP interpretation in the spirit of local ratio [18].)

Bar-Yehuda and Rawitz [12] presented local ratio interpretations of known algorithms for *minimum s-t cut* and the *assignment problem*. These algorithms are the first applications of local ratio to use negative weights. The corresponding primal-dual analyses are based on new IP formulations of these fundamental problems that contain negative coefficients.

Maximization problems. By the turn of the 20th century both methods were used extensively in the context of minimization algorithms. However, there was no application of either method that approximated a maximization problem. The first study to present a local-ratio and primal-dual approximation algorithm for a maximization problem was by Bar-Noy et al. [4]. In this paper the authors used the local-ratio technique to develop an approximation framework for resource allocation and scheduling problems. A primal-dual interpretation was also presented.

Bar-Noy et al.'s [4] result paved the way for other studies dealing with maximization problems. In [5] Bar-Noy et al. developed approximation algorithms for two variants of the problem of scheduling on identical machines with *batching*. Akcoglu at al. [2] presented approximation algorithms for several types of *combinatorial auctions*.

5.2 Equivalence Between the Two Methods

It has often been observed that primal-dual algorithms have local ratio interpretations, and vice versa. Bar-Yehuda and Even's primal-dual algorithm for *hitting set* [8] was analyzed using local ratio in [9]. The local ratio 2-approximation algorithm for *feedback vertex set* by Bafna et al. [3] was interpreted within the primal-dual schema [16]. The 2-approximation of a family of *network design* problems by Goemans and Williamson [21] was explained using local ratio in [6] (see also [7]). And, finally, Bar-Noy et al.'s [4] approximation framework for resource allocation and scheduling problems was developed initially using the local-ratio approach, and then explained it (in the same paper) in primal-dual terms. Thus, over the years there was a growing sense that the two seemingly distinct approaches share a common ground, but the exact nature of the connection between them remained unclear (see, e.g., [33], where this was posed as an open question). The issue was resolved in a paper by Bar-Yehuda and Rawitz [13], in which two approximation frameworks are defined, one encompassing the primal-dual schema, and the other encompassing the local ratio technique, and showed that these two frameworks are equivalent.

The equivalence between the paradigms is based on the simple fact that increasing a dual variable by ε is equivalent to subtracting the weight function obtained by multiplying the coefficients of the corresponding primal constraint by ε from the primal objective function. In other words, this means that an r-effective inequality can be viewed as an r-effective weight function and vice versa. For example, the coefficients of the generalized vertex cover constraint $\sum_{u \in S} x_u + x_S \geq 1$ are the same as the coefficients of the weight function w_1 from Algorithm **LR-GHS** up to a multiplitive factor of ε. Furthermore, both primal-dual analysis of Algorithm **PD-GHS** and local ratio analysis of Algorithm **LR-GHS** are based on Observation 6. The equivalence between the methods is constructive, meaning that an algorithm formulated within one paradigm can be translated quite mechanically to the other paradigm.

5.3 Fractional Local Ratio and Fractional Primal-Dual

The latest important development in the context of local ratio is a new variant of the local ratio technique called *fractional local ratio* [10]. As we have seen, a typical local ratio algorithm is recursive, and it constructs, in each recursive call, a new weight function w_1. In essence, a local ratio analysis consists of comparing, at each level of the recursion, the solution found in that level to an optimal solution for the problem instance passed to that level, where the comparison is made with respect to w_1. Thus, different optima are used at different recursion levels. The superposition of these "local optima" may be significantly worse than the "global optimum," i.e., the optimum of the original problem instance. Conceivably, we could obtain a better bound if at each level of the recursion we approximated the weight of a solution that is optimal with respect to the original weight function. This is the idea behind the fractional local ratio approach. More specifically, a fractional local ratio algorithm uses a single solution x^* to the original problem instance as the yardstick against which all intermediate solutions (at all levels of the recursion) are compared. In fact, x^* is not even feasible for the original problem instance but rather for a relaxation of it. Typically, x^* will be an optimal fractional solution to an LP relaxation of the problem.

Recently, Bar-Yehuda and Rawitz [14] have shown that the fractional approach extends to the primal-dual schema as well. As in fractional primal-dual the first step in a *fractional primal-dual* r-approximation algorithm is to compute an optimal solution to an LP relaxation of the problem. Let P be the LP relaxation, and let x^* be an optimal solution of P. Next, as usual in primal-dual algorithms, the algorithm produces an integral primal solution x and a dual solution y, such that r times the value of y bounds the weight of x (we use minimization terms). However, in contrast to other primal-dual algorithms, y is not a solution to the dual of P. The algorithm induces a new LP, denoted by P', that has the same objective function as P, but contains inequalities that may not be valid with respect to the original problem. Nevertheless, we make sure that x^* is a feasible solution of P'. The dual solution y is a feasible solution of the dual of P'. The primal solution x is r-approximate, since the optimum value of P' is not greater than the optimum value of P.

5.4 Further reading

A survey that describes the primal-dual schema and several recent extensions of the primal-dual approach is given in [33]. A detailed survey on the local ratio technique (including fractional local ratio) is given in [7].

Acknowledgment

We thank Oded Goldreich and Jonathan Laserson for their proof reading and suggestions.

References

1. A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

2. K. Akcoglu, J. Aspnes, B. DasGupta, and M.-Y. Kao. Opportunity cost algorithms for combinatorial auctions. In E. J. Kontoghiorghes, B. Rustem, and S. Siokos, editors, *Applied Optimization: Computational Methods in Decision-Making Economics and Finance*. Kluwer Academic Publishers, 2002.
3. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.
4. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and schedualing. *Journal of the ACM*, 48(5):1069–1090, 2001.
5. A. Bar-Noy, S. Guha, Y. Katz, J. Naor, B. Schieber, and H. Shachnai. Throughput maximization of real-time scheduling with batching. In *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 2002.
6. R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.
7. R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz. Local ratio: a unified framework for approximation algorithms. *ACM Computing Surveys*, 36(4):422–463, 2004.
8. R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.
9. 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.
10. R. Bar-Yehuda, M. M. Halldórsson, J. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. In *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 732–741, 2002.
11. R. Bar-Yehuda and D. Rawitz. Efficient algorithms for bounded integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001.
12. R. Bar-Yehuda and D. Rawitz. Local ratio with negative weights. *Operations Research Letters*, 32(6):540–546, 2004.
13. R. Bar-Yehuda and D. Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics*, 2005. To appear.
14. R. Bar-Yehuda and D. Rawitz. Using fractional primal-dual to schedule split intervals with demands. In *13th Annual European Symposium on Algorithms*, 2005. To appear.
15. 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.
16. 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.
17. V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
18. A. Freund and D. Rawitz. Combinatorial interpretations of dual fitting and primal fitting. In *1st International Workshop on Approximation and Online Algorithms*, volume 2909 of *LNCS*, pages 137–150. Springer-Verlag, 2003.
19. T. Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 86:213–231, 1998.
20. M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

21. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
22. M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Hochbaum [25], chapter 4, pages 144–191.
23. S. Guha, R. Hassin, S. Khuller, and E. Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003.
24. D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.
25. D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem*. PWS Publishing Company, 1997.
26. K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
27. 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.
28. H. Karloff. *Linear Programming*. Progress in Theoretical Computer Science. Birkhäuser Boston, 1991.
29. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization; Algorithms and Complexity*. Prentice-Hall, Inc., 5th edition, 1982.
30. R. Ravi and P. Klein. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *3rd Conference on Integer Programming and Combinatorial Optimization*, pages 39–56, 1993.
31. A. Schrijver. *Theory of linear and integer programming*. Wiley Publishers, 1998.
32. V. V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2nd edition, 2001.
33. D. P. Williamson. The primal dual method for approximation algorithms. *Mathematical Programming (Series B)*, 91(3):447–478, 2002.