The Primal-Dual Method for Approximation Algorithms

David P. Williamson*
IBM T.J. Watson Research Center and
IBM Almaden Research Center

Abstract

In this survey, we give an overview of a technique used to design and analyze algorithms that provide approximate solutions to NP-hard problems in combinatorial optimization. Because of parallels with the primal-dual method commonly used in combinatorial optimization, we call it the primal-dual method for approximation algorithms. We show how this technique can be used to derive approximation algorithms for a number of different problems, including network design problems, feedback vertex set problems, and facility location problems.

1 Introduction

Many problems of interest in combinatorial optimization are considered unlikely to have efficient algorithms; most of these problems are NP-hard, and unless $P = NP$ they do not have polynomial-time algorithms to find an optimal solution. Researchers in combinatorial optimization have considered several approaches to deal with NP-hard problems. These approaches fall into one of two classes. The first class contains algorithms that find the optimal solution but do not run in polynomial time. Integer programming is an example of such an approach. Integer programmers attempt to develop branch-and-bound (or branch-and-cut, etc.) algorithms for dealing with particular problems such that the algorithm runs quickly enough in practice for instances of interest, although the algorithm is not guaranteed to be efficient for all instances. The second class contains algorithms that run in polynomial time but do not find the optimal solution for all instances. Heuristics and metaheuristics (such as simulated annealing or genetic algorithms) are one approach in this class. Typically researchers develop a heuristic for a problem and empirically demonstrate its effectiveness on instances of interest. In this survey, we will consider another approach in this second class called *approximation algorithms*. Approximation algorithms are polynomial-time heuristics for NP-hard problems whose solution values are provably close to optimum for all instances of the problem.

More formally, an *α-approximation algorithm* for an optimization problem is an algorithm that runs in polynomial time and produces a solution whose value is within a factor of α of the value of an optimal solution. The parameter α is called the *performance guarantee* or the *approximation ratio* of the algorithm. We assume that the value of any feasible solution is nonnegative for the problems we consider; extensions of the notion of performance guarantee have been developed in other cases, but we will not discuss them here. This survey will follow the convention that $\alpha \geq 1$ for minimization problems and $\alpha \leq 1$ for maximization problems, so that a 2-approximation algorithm for a minimization problem produces a solution of value no more than twice the optimal value, and a $\frac{1}{2}$-approximation algorithm for a maximization problem produces a solution of value at least

*Current address: IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120. Email: dpw@almaden.ibm.com. WWW: www.research.ibm.com/people/w/williamson.

half the optimal value. The reciprocal $1/\alpha$ is sometimes used in the literature for maximization problems, so that the examples above would both be referred to as 2-approximation algorithms. We say that a minimization problem has a *polynomial-time approximation scheme* if there is a family $\{A_\epsilon\}$ of algorithms such that A_ϵ is a $(1+\epsilon)$-approximation algorithm for any fixed $\epsilon > 0$.

The term "approximation algorithm" appears to have been coined by David Johnson in a seminal paper in 1974 [51]. In this paper, Johnson gives approximation algorithms for several now-classical problems, including a polynomial-time approximation scheme for the subset sum problem, a $\frac{1}{2}$-approximation algorithm for the maximum satisfiability problem, an $(\ln n + 1)$-approximation algorithm for the set cover problem, as well as heuristics for graph coloring and maximum clique, though he could find no performance guarantee of $O(n^{1-\epsilon})$ for any $\epsilon > 0$ for the latter two problems, where n is the number of vertices in the graph. However, approximation algorithms were present in the literature even before the concept of NP-completeness was introduced. Lovász reports that a 1967 paper of Erdős [27] contains a proof that the size of maximum cut in a graph with non-negative edge weights is at least half the sum of the edge weights, and that the proof can easily be converted into a $\frac{1}{2}$-approximation algorithm for the maximum cut problem. Graham [44] gave 2-approximation algorithms for a number of scheduling problems in 1966. Finally, although it doesn't quite fit the definition given above of an approximation algorithm, in 1964 Vizing [68] gave an algorithm to compute an edge coloring of a graph which uses at most one color more than the minimum possible.

In the past dozen years there have been a number of exciting developments in the area of approximation algorithms. It is not possible in the space provided to give a comprehensive overview of these developments, so in this survey we will focus on one very useful algorithmic technique, called the primal-dual method, that has been developed and applied to several different problems in combinatorial optimization. However, we will very briefly touch on one other development in the following paragraphs. For more details about the area, the reader is invited to consult the excellent survey of Shmoys [65], the book of surveys edited by Hochbaum [48], or the monograph of Vazirani [67].

One very exciting development of the past decade is the emergence of proof techniques that show that many problems do not have approximation algorithms with certain performance guarantees unless $P = NP$. In other words, for some problems, finding an approximation algorithm with a particular performance guarantee is just as difficult as finding a polynomial-time algorithm for the problem itself. These results have their roots in research in theoretical computer science quite unrelated to optimization. The connection to optimization was made in a groundbreaking paper of Feige, Goldwasser, Lovász, Safra, and Szegedy [32], and culminated in two papers of Arora and Safra [4, 5] and Arora, Lund, Motwani, Sudan, and Szegedy [2, 3] which gave a new characterization of NP. As a consequence they showed that a collection of optimization problems could have no polynomial-time approximation scheme unless $P = NP$. This collection includes such problems as the symmetric traveling salesman problem with edge costs that obey the triangle inequality, the maximum cut problem, the minimum vertex cover problem, and the maximum satisfiability problem.

These results have subsequently been strengthened and extended in a number of different ways. For instance, a sequence of papers [32, 16], culminating in a paper of Håstad [46], has shown that for the maximum clique problem, no performance guarantee of $O(n^{1-\epsilon})$ for any $\epsilon > 0$ is possible unless $NP = RP$ (where n is the number of vertices in the graph, and RP is randomized polynomial time). Feige and Kilian extended this result to the minimum chromatic number problem [33]. Another sequence of papers, starting with the work of Lund and Yannakakis [57], and subsequently refined by other researchers [17, 31], has shown that there can be no $c \ln n$-approximation algorithm for the set cover problem for $c < 1$ unless there are $O(n^{O(\log \log n)})$-time algorithms for any NP-

complete problem. Still other papers have shown that one cannot obtain constant performance guarantees better than certain bounds for particular problems unless $P = NP$; for instance, $\frac{129}{128}$ for the symmetric traveling salesman problem with triangle inequality [61], $\frac{16}{17} \approx .941$ for the maximum cut problem [45], $\frac{7}{6}$ for the minimum-weight vertex cover problem [45], and $\frac{7}{8}$ for the maximum satisfiability problem [45]. It is interesting to note that several of the performance guarantees obtained in Johnson's original 1974 paper are shown to be essentially the best possible by these results.

We do not give an indepth review of the primal-dual method for combinatorial optimization here; a good overview can be found in the textbook of Papadimitriou and Steiglitz [60] (see also the survey of Goemans and Williamson [42]). The basic idea was first used by Kuhn [56] to give the so-called Hungarian algorithm for solving the assignment problem. It was then extended by Dantzig, Ford, and Fulkerson [22] to a general algorithm for linear programming. The basic idea is that given a feasible dual solution y, we attempt to find a feasible primal solution x that obeys the complementary slackness conditions with respect to y. If we can find such an x, we have optimal solutions. If none exists, this gives a way to modify the dual solution to increase the dual objective value. For combinatorial problems such as the assignment problem, the method converts weighted problems into unweighted ones. For instance, determining whether there exists a primal x obeying complementary slackness with respect to y in the case of the assignment problem reduces to finding the maximum matching in an unweighted bipartite graph on $2n$ nodes. Given the maximum matching it is easy to determine a direction for dual increase such that only $O(n^2)$ increases are necessary before we find optimal solutions.

The primal-dual method for approximation algorithms considers a primal integer programming formulation of the problem in question and the dual of a linear programming relaxation of the integer program. The method above is modified by relaxing complementary slackness conditions related to dual variables; that is, we relax the condition that if $y_j > 0$ the corresponding primal constraint must be met with equality. As we will see below, relaxing this constraint in appropriate ways leads to provably good algorithms for NP-hard problems in combinatorial optimization. The method yields a solution to the primal integer problem that costs no more than α times the value of the feasible dual solution constructed, which implies that the primal solution is within a factor of α of optimal. The value of the dual solution is always within some factor of α of the value of the primal solution, but may from instance to instance be much closer; by comparing the value of the primal and dual solutions generated, we can give a guarantee for the instance which might be better than α.

The performance guarantee of an algorithm using the primal-dual method is thus connected with the *integrality gap* of the integer programming formulation of the problem. The integrality gap of a formulation is the worst-case ratio over all instances of the value of the integer program to the value of the corresponding linear programming relaxation. Since the performance guarantee of an algorithm using the primal-dual method is proven by comparing the value of a primal solution against the value of a feasible dual, its performance guarantee can never be shown to be better than the integrality gap of the formulation used. Conversely, a proof of a performance guarantee of α obtained in this way implies that the integrality gap is no more than α.

So far the primal-dual method for approximation algorithms usually leads to dual-ascent algorithms in which dual variables are never decreased (though see Section 4 for some intriguing recent exceptions). Dual-ascent heuristics for hard combinatorial problems are not new; for example, see papers by Balakrishnan, Magnanti, and Wong [7], Erlenkotter [29], Wong [73], and the thesis of Raghavan [62]. However, such heuristics are not typically accompanied by performance guarantees, as is the case here.

As a brief illustration of the primal-dual method, we consider the *minimum-weight vertex cover* *problem*. In this problem, we are given a graph $G = (V, E)$ with weights $w_i \geq 0$ for all vertices $i \in V$, and we must select a minimum-weight subset of vertices such that each edge is covered (that is, at least one of its endpoints is chosen). This problem can be modelled by the integer program

$$\begin{aligned} \text{Min} \quad & \sum_{i \in V} w_i x_i \\ \text{subject to:} \quad & \\ & x_i + x_j \geq 1 \qquad && \forall (i,j) \in E \\ & x_i \in \{0,1\} && \forall i \in V. \end{aligned}$$

We relax the integrality constraint $x_i \in \{0,1\}$ to $x_i \geq 0$; any optimal solution x^* to this LP will have $x_i^* \leq 1$ for all $i \in V$. If we take the dual of the resulting linear program, we obtain the following:

$$\begin{aligned} \text{Max} \quad & \sum_{(i,j) \in E} y_{(i,j)} \\ \text{subject to:} \quad & \\ & \sum_{k:(i,k) \in E} y_{(i,k)} \leq w_i \qquad && \forall i \in V \qquad (1) \\ & y_{(i,j)} \geq 0 && \forall (i,j) \in E. \end{aligned}$$

Our primal-dual algorithm starts out with the dual feasible solution in which all y variables are set to 0, and a primal infeasible solution in which all x variables are set to 0. As long as our primal solution x is infeasible, there must be some uncovered edge (i,j) for which $x_i + x_j = 0$. We increase its corresponding dual variable $y_{(i,j)}$ as much as possible, maintaining dual feasibility, so that it must be the case that the dual constraint (1) is met with equality for either i or j (possibly both). If $\sum_{k:(i,k) \in E} y_{(i,k)} = w_i$, we set $x_i = 1$ and if $\sum_{k:(j,k) \in E} y_{(j,k)} = w_j$ we set $x_j = 1$. Eventually we achieve a primal feasible solution x such that

$$\begin{aligned} \sum_{i \in V} w_i x_i &= \sum_{i \in V} \left(\sum_{k:(i,k) \in E} y_{(i,k)} \right) x_i && (2) \\ &= \sum_{(i,j) \in E} (x_i + x_j)\, y_{(i,j)} && (3) \\ &\leq 2 \sum_{(i,j) \in E} y_{(i,j)}, && (4) \end{aligned}$$

where the equality (2) follows since $\sum_{k:(i,k) \in E} y_{(i,k)} = w_i$ for all $x_i = 1$, the equality (3) follows by rearranging the double sum, and the inequality (4) follows since $x_i + x_j \leq 2$. The dual objective function $\sum_{(i,j) \in E} y_{(i,j)}$ is a lower bound on the value of the optimum integer solution. Thus the inequality above shows that our solution is no more than twice optimal, implying that the algorithm is a 2-approximation algorithm.

In the next section, we develop the basic ideas given above into a primal-dual algorithm for a generic problem, and give theorems for its analysis. In Section 3, we give applications of this algorithm and analysis to various NP-hard problems in combinatorial optimization. Then in Section 4, we show how recent papers have modified this central algorithm to obtain new approximation algorithms for other problems. We conclude in Section 5.

Other surveys on the primal-dual method have been given by Goemans and Williamson [42] and Bertsimas and Teo [18] (see also the thesis of Teo [66]). Our central exposition in Section 2 closely follows that of [42].

2 The primal-dual method for approximation algorithms

We now show how the primal-dual method can be used to give approximation algorithms for NP-hard problems in combinatorial optimization. In order to do this, it will be useful to consider the *hitting set problem*: given a ground set of elements E, nonnegative costs c_e for all elements $e \in E$, and subsets $T_1, \ldots, T_p \subseteq E$, we want to find a minimum-cost subset $A \subseteq E$ so that A has a nonempty intersection with each subset T_i. We say that A *hits* each subset T_i.

The hitting set problem can be used to model a number of NP-hard problems, and we will consider several in this section. For example, we can formulate the minimum-weight vertex cover problem as a hitting set problem in which the ground set elements are vertices, and we have a subset $T_i = \{u, v\}$ for each edge (u, v) in the graph. In the *minimum-weight feedback vertex set problem in undirected graphs*, we are given as input an undirected graph $G = (V, E)$ and nonnegative weights $w_i \geq 0$ on the vertices $i \in V$, and the goal is to remove a minimum-weight set of vertices from G so as to make the remaining graph acyclic. We can view this as a hitting set problem in which the ground set elements are the vertices of the graph, and we must hit every cycle in the graph; that is, $T_i = C_i$, where C_i is the ith cycle of G. In the *shortest s-t path problem*, we are given an undirected graph with nonnegative edge costs c_e for all $e \in E$, and two distinguished vertices s and t, and we must find the minimum-cost path from s to t. We can formulate this as a hitting set problem in which the edges are the ground set elements and we must hit every cut in the graph separating s from t; that is, for all $S_i \subseteq V$ with $s \in S_i$ and $t \notin S_i$, we must select an edge from $T_i = \delta(S_i)$, where $\delta(S_i)$ is the set of edges with exactly one endpoint in S_i. By the max-flow/min-cut theorem of Ford and Fulkerson [34], we have selected an edge in every cut separating s from t iff there is a path from s to t. In the *minimum-cost branching problem* we are given a directed graph $G = (V, A)$, nonnegative costs c_a for all arcs $a \in A$, and a root vertex $r \in V$, and the goal is to find a minimum-cost branching (a set of arcs such that for every vertex, there is a path from the root to the vertex). By using a max-flow/min-cut argument, one can see that the following hitting set problem models the minimum-cost branching problem: the ground set of elements are the arcs, and for every set of vertices $S_i \subseteq V - r$, we must hit the set $\delta^-(S_i)$ of arcs, where $\delta^-(S_i)$ is the set of arcs whose heads are in S_i and tails are not in S_i. Finally, in the *generalized Steiner tree problem* we are given an undirected graph $G = (V, E)$, nonnegative costs $c_e \geq 0$ on all edges $e \in E$, and k pairs of vertices $s_j, t_j \in V$. The goal is to find a minimum-cost set of edges F, such that for each $j = 1, \ldots, k$, s_j and t_j are connected in the graph (V, F). Again, a max-flow/min-cut argument will show that the problem can be modelled by the hitting set problem in which the ground set elements are the edges and we must hit every cut that separates some s_j-t_j pair; in other words, for each S_i such that for some j, $|S_i \cap \{s_j, t_j\}| = 1$, we must hit $T_i = \delta(S_i)$.

Except for the minimum-cost s-t path problem and the minimum-cost branching problem, all of the problems above are NP-hard. For many of them, the size of the hitting set formulation is exponential in the size of the input. For example, in the feedback vertex set problem, the number of cycles can be exponential in the size of the graph. We will see later that the primal-dual method can often be used in these cases and still results in a polynomial-time algorithm.

We can model the hitting set problem by the following integer program:

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{e \in T_i} x_e \geq 1 \qquad \forall i$$

$$x_e \in \{0, 1\} \qquad \forall e \in E.$$

We relax the integrality constraint $x_e \in \{0, 1\}$ to $x_e \geq 0$; as in the case of the vertex cover problem, any optimal solution x^* to this linear program will have $x^*_e \leq 1$ for all $e \in E$. If we take the dual of the resulting linear program, we obtain the following:

$$
\begin{aligned}
& \text{Max} \quad \sum_{i=1}^{p} y_i \\
\text{subject to:} & \\
& \sum_{i: e \in T_i} y_i \leq c_e \qquad && \forall e \in E \\
& y_i \geq 0 && \forall i.
\end{aligned}
$$

Our goal is to construct a feasible solution $\bar{x}$ to the primal integer program and a feasible solution y to the dual linear program such that $\sum_{e \in E} c_e \bar{x}_e \leq \alpha \cdot \sum_{i=1}^{p} y_i$ for some value of α. This implies that the cost of our primal solution is no more than α times the cost of an optimal solution to the integer program. If we can construct our solutions in polynomial time, then we have an α-approximation algorithm. We will sometimes give our primal solution as $\bar{x}$ or as a subset $A \subseteq E$, which implies the solution $\bar{x}_e = 1$ for $e \in A$ and $\bar{x}_e = 0$ otherwise.

The development of the primal-dual method for approximation algorithms can be said to start with a non-primal-dual approximation algorithm for the hitting set problem due to Hochbaum [47]. Hochbaum's algorithm obtains an optimal solution y^* to the dual LP, and then constructs a primal solution A by choosing all elements $e \in E$ such that the corresponding dual inequality is *tight* (that is, met with equality). So then $A = \{e \in E : \sum_{i: e \in T_i} y^*_i = c_e\}$. In terms of the primal-dual method for combinatorial optimization, we construct a primal solution such that the complementary slackness conditions are obeyed with respect to the primal variables x_e; that is, $x_e > 0$ implies that $\sum_{i: e \in T_i} y^*_i = c_e$. We claim that A is a feasible solution to the hitting set problem, and we will prove this later. Then the cost of this solution is

$$
\begin{aligned}
\sum_{e \in A} c_e &= \sum_{e \in A} \sum_{i: e \in T_i} y^*_i && (5) \\
&= \sum_{i=1}^{p} y^*_i |A \cap T_i|, && (6)
\end{aligned}
$$

where (5) follows since the complementary slackness conditions are obeyed for the primal variables, and (6) follows by reversing the double sum. If we let $f = \max_i |T_i|$, then certainly $|A \cap T_i| \leq f$ for all i, so that

$$\sum_{e \in A} c_e \leq f \cdot \sum_{i=1}^{p} y^*_i.$$

Assuming the claim that A is feasible, we thus have an f-approximation algorithm for the hitting set problem. As an example of what can be proved in this case, recall that in the minimum-weight vertex cover problem each subset T_i contained the two endpoints of an edge in a graph, so that $|T_i| = 2$ for each i in this case. Thus Hochbaum's algorithm gives a 2-approximation algorithm for the minimum-weight vertex cover problem.

We now prove the claim of feasibility. The central idea, as in the case of the standard primal-dual method, is that if no primal solution obeys the complementary slackness conditions, then a dual increase is possible.

Lemma 1 (Hochbaum [47]) *The set A obtained above is a feasible solution for the hitting set problem.*

1 $y \leftarrow 0$
2 $A \leftarrow \emptyset$
3 While A is not feasible
4   Find violated T_k (i.e. T_k s.t. $A \cap T_k = \emptyset$)
5   Increase y_k until $\exists\ e \in T_k$ such that $\sum_{i:e \in T_i} y_i = c_e$
6     $A \leftarrow A \cup \{e\}$
7 Return A.

Figure 1: The basic primal-dual algorithm.

Proof: Suppose not. Then there is some set T_i such that $A \cap T_i = \emptyset$. By the choice of A, it follows that for all $e \in T_i$, $\sum_{i:e \in T_i} y_i^* < c_e$. Since the dual inequalities for $e \in T_i$ are the only ones in which the variable y_i^* participates, we can feasibly increase y_i^* by $\epsilon > 0$, where $\epsilon = \min_{e \in T_i} \left(c_e - \sum_{k:e \in T_k} y_k^* \right)$. This contradicts the optimality of y^*. □

The first algorithm using the primal-dual method for approximation algorithms is due to Bar-Yehuda and Even [12]. Essentially, they realized that an optimal dual solution y^* was not needed; the proof above goes through for any feasible dual solution y such that $A = \{e \in E : \sum_{i:e \in T_i} y_i = c_e\}$ is a feasible solution to the hitting set problem. Furthermore, if A is not feasible, then the proof of Lemma 1 shows how to improve a current dual feasible solution so that the dual objective function increases, and so that there is one more tight dual inequality. Thus given an initial dual feasible solution, this process gives a polynomial-time algorithm that eventually finds a feasible dual solution y and a feasible primal solution $A = \{e \in E : \sum_{i:e \in T_i} y_i = c_e\}$; by the reasoning above, $\sum_{e \in A} c_e \leq \sum_{i=1}^{p} y_i |A \cap T_i| \leq f \sum_{i=1}^{p} y_i$. Since all costs c_e are non-negative, the dual solution $y_i = 0$ for all i can be used as an initial feasible dual solution. We summarize Bar-Yehuda and Even's algorithm in Figure 1. The argument above shows that it is an f-approximation algorithm for the hitting set problem. The 2-approximation algorithm for the vertex cover problem given in the introduction is this algorithm specialized to the vertex cover problem.

We now turn to a slightly more complicated application of the primal-dual algorithm: the feedback vertex set problem for undirected graphs. Let A and y be the primal and dual solution created by the algorithm. Recall from equations (5) and (6) that if for any $y_i > 0$ it is the case that $|A \cap T_i| \leq \alpha$, then the algorithm is an α-approximation algorithm. Recall now that for the hitting set problem modelling this problem, each ground element e is a vertex j, the cost c_e is the vertex weight w_j, and the sets T_i are the cycles in the graph. In this case, Bar-Yehuda, Naor, Geiger, and Roth [14] obtain a performance guarantee of $4 \log_2 n$ (where $n = |V|$) by carefully choosing the violated cycle in line 4 of the algorithm, and by noticing that one can succesfully ignore some vertices since their corresponding dual inequalities will always be satisfied. In order to choose the violated cycle, Bar-Yehuda et al. invoke the following lemma of Erdős and Pósa [28].

Lemma 2 (Erdős and Pósa [28]) *Given a graph $G' = (V', E')$ with no degree 1 vertices and with every vertex of degree 2 adjacent to two vertices of higher degree, there exists a cycle of length no longer than $4 \log_2 |V'|$, and it can be found in polynomial time.*

Of course, the given input graph might not meet the conditions of the lemma. Thus we show that we can ignore some vertices; the remaining vertices we call *special* vertices. We map the graph onto a graph G' that contains exactly the special vertices, such that there is a bijective mapping between cycles of G and of G'. Then by applying the lemma to G' we can find in G a violated cycle of at most $4\log_2 n$ special vertices, and since we only add special vertices to A, we get that for any $y_i > 0$ (corresponding to some violated cycle T_i chosen in line 4), $|A \cap T_i| \leq 4\log_2 n$, implying the desired performance guarantee.

Now we need to specify which vertices we can ignore, and why their dual inequalities will remain feasible. Suppose that as we add a vertex j to A in line 6 of the algorithm, we remove j and its incident edges from the graph. Certainly we can ignore any vertex in the remaining graph that is no longer in a cycle; since we only add vertices from the chosen violated set (line 5), we only add vertices that are in some cycle. Now consider any path of vertices that all have degree 2. Since any cycle that goes through one of these vertices must go through all of them, it must be the case that when the reduced cost $\tilde{w}_j = w_j - \sum_{i:j\in T_i} y_i$ of a vertex j on this path decreases by ϵ, the reduced cost of all vertices on this path also decreases by ϵ. Thus we can safely ignore all vertices in this path except for one special vertex j with the smallest reduced cost, since no dual inequality for any vertex on the path will become tight unless the dual inequality for j becomes tight. Furthermore, if j is added to A, then all cycles containing the vertices on this path will be hit, and so no other vertex from the path need by added to A.

Since we can ignore any vertex not on a cycle, and ignore all but one vertex on a path of vertices of degree 2, we obtain the desired graph G' from G by removing all vertices currently in A, recursively removing all degree 1 vertices, and replacing any path of degree 2 vertices with the special vertex for that path. This yields a graph G' obeying the properties of the lemma, such that any cycle in G' has a one-to-one mapping to a cycle of G. Thus we can find a cycle of at most $4\log_2 n$ special vertices in G.

This argument yields a $(4\log_2 n)$-approximation algorithm for the minimum-weight feedback vertex set problem in undirected graphs. In fact, as we will discuss in Section 3.2, one can obtain a 2-approximation algorithm for this problem using the primal-dual method, but one must use a different integer programming formulation of the problem. It has been shown that the integrality gap for the hitting set formulation of the problem is $\Omega(\log n)$ [30].

We now turn to modifications of the basic primal-dual algorithm of Bar-Yehuda and Even. The first is a relatively simple idea: once a feasible solution A has been obtained, we should examine the elements of A and delete any that are not needed for a feasible solution. This idea was first introduced by Goemans and Williamson [41], but we will present here a refinement discovered independently by Klein and Ravi [53] and Saran, Vazirani, and Young [64]. They showed that it is useful for the analysis of the algorithm to examine the elements of A for possible deletion in a certain order; in particular, in the reverse of the order in which the elements of A were added. This part of the algorithm is sometimes called the *reverse delete*. We present the modified algorithm in Figure 2.

To see why the reverse delete step is useful for the analysis, consider the set T_{i_l} chosen in the lth iteration of the algorithm. Let A_l be the set of elements in A at the beginning of the lth iteration, let e_l be the element added in the lth iteration, and let A' be the final set returned by the algorithm. By the analysis at the beginning of the section (Equations (5) and (6)), if we can show that $|A' \cap T_{i_l}| \leq \alpha$ for all iterations l, we have an α-approximation algorithm. Note that since T_{i_l} is chosen as a violated set, it is the case that $T_{i_l} \cap A_l = \emptyset$, so if $B = A' - A_l$, then we only need prove that $|B \cap T_{i_l}| \leq \alpha$. Furthermore, when e_l is considered for deletion, no element e_j for $j < l$ has been considered for deletion, so the contents of A at that point in time in the reverse delete step must be precisely $A_l \cup B$. Finally, because each element in B was added after the lth iteration, it must be the case that each of them was already considered by the reverse delete step and is necessary for the feasibility of $A_l \cup B$. Thus for any $e \in B$, $A_l \cup B - e$ is not a feasible solution. We call any set of elements D such that $A_l \cup D$ is feasible an *augmentation* of A_l, and

$y \leftarrow 0$
$A_1 \leftarrow \emptyset$
$l \leftarrow 1$ (l is a counter)
While A_l is not feasible
    Choose violated T_k
    Increase y_k until $\exists\ e_l \in T_k$ such that $\sum_{i:e_l \in T_i} y_i = c_{e_l}$
    $A_{l+1} \leftarrow A_l \cup \{e_l\}$
    $l \leftarrow l + 1$
$A' \leftarrow A_{l-1}$
For $j \leftarrow l - 1$ down to 1
    If $A' - \{e_j\}$ is still feasible
        $A' \leftarrow A' - \{e_j\}$
Return A'.

Figure 2: The primal-dual algorithm with reverse delete step added.

any augmentation D such that for any $e \in D$, $A_l \cup D - e$ is not feasible, a *minimal* augmentation. We have shown above that B is a minimal augmentation of A_l. We are trying to bound $|B \cap T_{i_l}|$, and certainly this is dominated by the maximum of $|D \cap T_{i_l}|$ over all minimal augmentations D of A_l. Thus we have shown the following theorem.

Theorem 3 *If for all iterations l of the algorithm in Figure 2,*

$$\max_{D:\text{min. aug. of } A_l} |D \cap T_{i_l}| \leq \alpha,$$

the algorithm is an α-approximation algorithm.

To illustrate the use of this analysis, we consider the shortest s-t path problem and the minimum-cost branching problem. Recall that for the minimum-cost s-t path problem, we need to hit the sets $T_i = \delta(S_i)$ for all sets S_i with $s \in S_i$, $t \notin S_i$, where $\delta(S_i)$ is the set of edges with exactly one endpoint in S_i. To apply the primal-dual algorithm of Figure 2, we need to specify which violated set T_{i_l} is chosen for a given infeasible solution A_l. Here we invoke a principle that turns out to be useful for a number of problems of this sort: we choose the *minimal* violated set $T_i = \delta(S_i)$, where by this we mean a set S_i such that there is no other set $S_j \subset S_i$ with $T_j = \delta(S_j)$ also violated. For the minimum-cost s-t path problem, this principle implies that for an infeasible solution A_l, we find the connected component S_{i_l} containing s in the graph (V, A_l), and choose the violated set $T_{i_l} = \delta(S_{i_l})$. It is not difficult to see that for any augmentation D of A_l, if $|D \cap \delta(S_{i_l})| > 1$, then an edge of $D \cap \delta(S_{i_l})$ can be removed with the remaining edges still containing an s-t path. Thus for any minimal augmentation D, it is the case that $|D \cap \delta(S_{i_l})| = 1$, which implies by the analysis of the preceding paragraph that the primal-dual method gives a 1-approximation algorithm, or an optimal algorithm, for the shortest s-t path problem. In fact, one can show that this algorithm is just Dijsktra's algorithm [23, 69].

For the minimum-cost branching problem, we need to hit the sets $T_i = \delta^-(S_i)$ for all $S_i \subseteq V - r$. Recall that $\delta^-(S_i)$ is the set of arcs with their heads in S_i and their tails not in S_i. Given an infeasible set A_l, we find a strongly connected component S_{i_l} in the graph (V, A_l) which does not contain the root r and for which $A_l \cap \delta^-(S_{i_l}) = \emptyset$. We choose as our violated set $T_{i_l} = \delta^-(S_{i_l})$. It is not hard to show that such a strongly connected component must exist if A_l is infeasible. Then again it is easy to see that for any augmentation D of A_l, only one arc in $D \cap \delta^-(S_{i_l})$ is necessary, since the strong connectivity of S_{i_l} implies all the vertices of S_{i_l} can be reached through that arc. Hence for any minimal augmentation D of A_l, $|D \cap \delta^-(S_{i_l})| = 1$, and we again have an optimal algorithm. One can show that this algorithm is the same as Edmonds' algorithm for the minimum-cost branching problem [25].

We now introduce another modification to our primal-dual algorithm. To motivate the modification, we consider the generalized Steiner tree problem. Recall that this can be modelled by a hitting set problem in which we must hit all $T_i = \delta(S_i)$ such that $|S_i \cap \{s_j, t_j\}| = 1$ for some s_j-t_j pair that must be connected. Suppose we try to apply the algorithm in Figure 2 and the analysis above to this problem. As with the shortest s-t path problem, we will invoke the principle of finding a minimal violated set and choose some connected component S_{i_l} of (V, A_l) such that $|S_{i_l} \cap \{s_j, t_j\}| = 1$, and choose as our violated set $T_{i_l} = \delta(S_{i_l})$. However, consider the problem in which $s = s_1 = s_2 = \cdots = s_k$, and $t_1, \ldots, t_k$ are distinct vertices. Then for $A_1 = \emptyset$, the vertex s and each t_j is a possible minimal violated set. Without loss of generality, suppose we choose the violated set $T = \delta(\{s\})$. Then one possible minimal augmentation is $D = \{(s, t_1), (s, t_2), \ldots, (s, t_k)\}$, and $|D \cap T| = k$. Thus the algorithm and analysis we have developed so far would only give a k-approximation algorithm.

However, if we consider the number of times this augmentation hits these minimal violated sets averaged over the number of minimal violated sets, we get something better: $|D \cap \delta(\{s\})| = k$, but $|D \cap \delta(\{t_j\})| = 1$, with $k+1$ minimal violated sets, leading to an average of $2k/(k+1) \approx 2$. This leads to the following idea: suppose we choose multiple violated sets and increase their dual variables simultaneously and uniformly. It turns out that this gives good approximation algorithms for a number of problems, including a 2-approximation algorithm for the generalized Steiner tree problem. We give the modified algorithm in Figure 3. The idea of increasing multiple duals was introduced implicitly by Agrawal, Klein, and Ravi [1] (who did not refer to LP duality), and was made explicit by Goemans and Williamson [41]. Bertsimas and Teo [18] reduce this algorithm to the one in Figure 2 by observing that the constraints corresponding to the multiple violated sets chosen in a given iteration can be aggregated into a single constraint, whose corresponding dual can be increased as in the previous algorithm in Figure 2.

We now show how we can analyze the algorithm in Figure 3 via the following theorem. Notice that this algorithm and its analysis generalize the algorithm of Figure 2, in which only one violated set is chosen in each iteration.

Theorem 4 *If for every iteration l of the algorithm in Figure 3,*

$$\max_{D:\text{min. aug. of } A_l} \sum_{T_k \in \mathcal{V}_l} |D \cap T_k| \leq \alpha |\mathcal{V}_l|,$$

the algorithm is an α-approximation algorithm.

Proof: Let A' be the final solution returned by the algorithm. We wish to prove that $\sum_{e \in A'} c_e \leq \alpha \sum_{i=1}^p y_i$. As before, we have that

$$\sum_{e \in A'} c_e = \sum_{e \in A'} \sum_{i: e \in T_i} y_i = \sum_{i=1}^p |A' \cap T_i| y_i.$$

So we need to prove that

$$\sum_{i=1}^p |A' \cap T_i| y_i \leq \alpha \sum_{i=1}^p y_i.$$

$y \leftarrow 0$
$A_1 \leftarrow \emptyset$
$l \leftarrow 1$ (l is a counter)
While A_l is not feasible
    Choose a subset $\mathcal{V}_l$ of violated sets
    Increase y_k uniformly for all $T_k \in \mathcal{V}_l$ until $\exists\ e_l \notin A_l$
        such that $\sum_{i:e_l \in T_i} y_i = c_{e_l}$
    $A_{l+1} \leftarrow A_l \cup \{e_l\}$
    $l \leftarrow l+1$
$A' \leftarrow A_{l-1}$
For $j \leftarrow l-1$ down to 1
    If $A' - \{e_j\}$ is still feasible
        $A' \leftarrow A' - \{e_j\}$
Return A'.

Figure 3: The general primal-dual algorithm.

Let ϵ_l be the amount by which the duals are increased in iteration l of the algorithm. Then clearly, for the solution y at the end of the algorithm,

$$\sum_{i=1}^{p} y_i = \sum_l |\mathcal{V}_l| \epsilon_l.$$

Similarly,

$$\sum_{i=1}^{p} |A' \cap T_i| y_i = \sum_{i=1}^{p} |A' \cap T_i| \sum_{l:T_i \in \mathcal{V}_l} \epsilon_l = \sum_l \left(\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k| \right) \epsilon_l.$$

Thus certainly the inequality follows if for all iterations l,

$$\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k| \leq \alpha |\mathcal{V}_l|.$$

As in the proof of Theorem 3, $\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k|$ is dominated by $\max_{D\text{:min. aug. of } A_l} \sum_{T_k \in \mathcal{V}_l} |D \cap T_k|$. Thus the theorem follows. □

To illustrate the use of the algorithm and the theorem, we show how we can obtain a 2-approximation algorithm for the generalized Steiner tree problem. As suggested above, in each iteration l we choose all the minimal violated sets; that is, we choose the sets $T_i = \delta(S_i)$ for all connected components S_i in (V, A_l) such that for some j, S_i contains exactly one of s_j or t_j. Thus

$$\mathcal{V}_l = \{T_i = \delta(S_i) : S_i \text{ a connected component of } (V, A_l), |S_i \cap \{s_j, t_j\}| = 1 \text{ for some } j\}.$$

Theorem 5 *Using the algorithm in Figure 3 with the choice of $\mathcal{V}_l$ as given above yields a 2-approximation algorithm for the generalized Steiner tree problem.*

Proof: To prove this, we show that the statement of Theorem 4 holds for $\alpha = 2$. To do this, we consider the graph in which each connected component of (V, A_l) has been shrunk to a single node; let V' be this set of vertices. Let D be any minimal augmentation of A_l, and consider the graph

$H = (V', D)$. Note first that H is a forest, otherwise D is not minimal. Observe also that some of the vertices in V' correspond to connected components S_i that are in $\mathcal{V}_l$ and some do not. Let $R \subseteq V'$ be the first type of vertex, which we will call red, and $B = V' - R$ be the second type, which we will call blue. Observe that $|R| = |\mathcal{V}_l|$. Also, if $deg(v)$ is the degree of $v \in V'$ in the graph H, and v corresponds to the connected component S_i in (V, A_l), then $|D \cap T_i| = |D \cap \delta(S_i)| = deg(v)$. Thus the desired inequality

$$\sum_{T_k \in \mathcal{V}_l} |D \cap T_k| \leq 2|\mathcal{V}_l|$$

reduces to proving that $\sum_{v \in R} deg(v) \leq 2|R|$. If we can show that no blue vertex has degree 1, then the statement would follow, since (ignoring blue vertices of degree 0),

$$\begin{aligned} \sum_{v \in R} deg(v) &= \sum_{v \in R \cup B} deg(v) - \sum_{v \in B} deg(v) \\ &\leq 2(|R| + |B|) - 2|B| \\ &= 2|R|. \end{aligned}$$

The inequalities follow since the sum of degrees of the vertices in the forest H is no more than twice the number of vertices, and every blue vertex in the sum has degree at least 2. To show that no blue vertex has degree 1, assume the opposite: let v be a blue vertex of degree 1, let $e \in D$ be the adjacent edge in H, and let S be the connected component corresponding to v in (V, A_l). Because D is a minimal augmentation, e is necessary for feasibility. Since e is the only edge in $D \cap \delta(S)$ there must be some j such that either s_j or t_j is in S and the other is not in S. But then $T = \delta(S)$ would be in $\mathcal{V}_l$, and v would be red, which is a contradiction. □

Thus the algorithm in Figure 3 gives a 2-approximation algorithm for the generalized Steiner tree problem. The first 2-approximation algorithm for this problem was given by Agrawal, Klein, and Ravi [1]. Its use of the primal-dual method was made explicit by Goemans and Williamson [41].

Approximation algorithms for many NP-hard problems can be derived from the framework above, as we will see in the following section. However, it is important to remember that the algorithm and analysis given above is only one potential way of applying the primal-dual technique, the one that developed historically from papers in the 80s and early 90s. A few recent papers have used their own variations of the primal-dual method; we will discuss these in Section 4.

3 Some applications of the primal-dual framework

In this section, we describe some of the results that can be obtained directly from the algorithm of Figure 3 and its analysis in Theorem 4.

3.1 Network design problems

The first application of the primal-dual algorithm of Figure 3 was to network design problems. It was first applied (implicitly) to the generalized Steiner tree problem by Agrawal, Klein, and Ravi [1], and was then generalized to apply to a number of other network design problems by Goemans and Williamson [41, 40], Klein and Ravi [53], Williamson, Goemans, Mihail, and Vazirani [71], Gabow, Goemans, and Williamson [36], and Goemans, Goldberg, Plotkin, Shmoys, Tardos, and Williamson [39]. This line of work is summarized in the survey of Goemans and Williamson [42].

Given an undirected graph $G = (V, E)$ and nonnegative costs c_e for all $e \in E$, consider the following integer programming formulation:

$$\begin{array}{llll} & \text{Min} & \sum_{e \in E} c_e x_e & \\ & \text{subject to:} & & \\ (ND) & & \sum_{e \in \delta(S)} x_e \geq f(S) & \forall S : \emptyset \neq S \subset V \\ & & x_e \in \{0,1\}, & \end{array}$$

where $f : 2^V \rightarrow \{0,1\}$. This integer program corresponds to a hitting set problem in which the edge set E is the set of ground elements, and we must hit all sets $T = \delta(S)$ for which $f(S) = 1$. Clearly this models the generalized Steiner tree problem in the case that $f(S) = 1$ iff $|S \cap \{s_j, t_j\}| = 1$ for some j.

In fact, the integer program can be used to model a number of network design problems. It models the minimum spanning tree problem when $f(S) = 1$ for all $S \subseteq V$, $S \neq \emptyset$. In the *Steiner tree problem*, we are given a set of terminals $T \subseteq V$, and must return a minimum-cost tree connecting all the vertices in T. The integer program models this problem when $f(S) = 1$ iff $0 < |S \cap T| < |T|$; that is, the cut S separates a pair of terminals. It models the shortest s-t path problem when $f(S) = 1$ iff $|S \cap \{s, t\}| = 1$. The T-join problem is one of finding a minimum-cost forest such that all vertices in T have odd degree and all other vertices have even degree (clearly, $|T|$ must be even). This can be modelled by the integer program when $f(S) = 1$ iff $|S \cap T|$ is odd. Consider the problem of finding a minimum-cost set of edges such that every connected component has $0 (\text{mod } k)$ vertices for any k such that $|V| \equiv 0 (\text{mod } k)$. We call this a tree partitioning problem, and it can be modelled by the integer program by setting $f(S) = 1$ iff $|S| \not\equiv 0 (\text{mod } k)$.

All of the functions f used for these problems are *proper* functions. We say a function $f : 2^V \rightarrow N$ is proper if $f(V) = 0$, $f(S) = f(V - S)$ for all $S \subseteq V$, and for any disjoint A and B, $f(A \cup B) \leq \max(f(A), f(B))$. Goemans and Williamson [41] show that the algorithm in Figure 3 gives a 2-approximation algorithm for (ND) for any proper function with range $\{0, 1\}$. To apply the algorithm, they use the principle of finding minimal violated sets, which for any infeasible solution A is the set of connected components C of (V, A) such that $f(C) = 1$. That is, in every iteration l of the algorithm $\mathcal{V}_l = \{T_i = \delta(S_i) : S_i \text{ a connected component of } (V, A_l), f(S_i) = 1\}$. We can apply Theorem 4 with $\alpha = 2$ to show that this gives us a 2-approximation algorithm. In fact, the proof of this is almost identical to the proof above of Theorem 5 for the generalized Steiner tree problem. We only need to modify the last part of the proof to show that for any proper function, no blue node has degree 1; we leave this as an exercise for the reader. This algorithm can be implemented in $O(n^2 \log n)$ time for these problems using simple data structures, and somewhat faster running times by using more complicated data structures (see Klein [54], Gabow et al. [36]).

Thus we get a number of 2-approximation algorithms for various problems. For the minimum spanning tree, we get an optimal algorithm, since in this case the primal-dual algorithm emulates Kruskal's algorithm [55]. In the case of the Steiner tree problem, we get an algorithm which emulates a number of previously known 2-approximation algorithms for the Steiner tree problem (see the survey of Winter [72]). In the case of the shortest s-t path problem, we get an optimal algorithm. In the case of the T-join problem, a polynomial-time algorithm is known, due to Edmonds and Johnson [26]. The primal-dual algorithm gives a 2-approximation algorithm with a running time faster than the best known running time of the Edmonds-Johnson algorithm on dense graphs. And in the case of the tree partitioning problem, we get a 2-approximation algorithm. In fact, given that edge costs obey the triangle inequality, we can use this algorithm to get approximation algorithms for a number of other problems. By partitioning the graph into trees of even size (using $k = 2$), we can obtain a matching of the graph by doubling each tree, shortcutting the tree to a tour of its vertices, then choosing the cheaper of the two matchings imposed by the tour. This gives us a 2-approximation algorithm of the minimum-cost perfect matching problem whose running time is faster than the best known matching algorithm on dense graphs. Williamson and Goemans [70] have implemented the matching algorithm and found that it is typically within 4% of optimal.

In a similar manner, we can get 2-approximation algorithms in the case that the function $f : 2^V \to \{0,1\}$ is *downwards monotone* [40]. We say that f is downwards monotone if $f(S) \leq f(T)$ for all $S \supseteq T \neq \emptyset$. We can use this to model the problem of partitioning the graph into trees each of which has at least k vertices (with $f(S) = 1$ if $0 < |S| < k$), and some location-design and location-routing problems [40].

The most sophisticated use of the primal-dual method for network design problems is an approximation algorithm that works for any proper function f. One problem that can be modelled by (ND) with such a function is the *survivable network design problem* (SNDP). In this problem, a value r_{ij} is given for every pair of nodes i, j, and one must find a minimum-cost set of edges such that for every i, j pair there are at least r_{ij} edge-disjoint paths between i and j. This problem arises in the design of low-cost fault-tolerant networks, since it implies that i and j will still be connected even after $r_{ij} - 1$ edge failures. By using the function $f(S) = \max_{i \in S, j \notin S} r_{ij}$, the integer program (ND) models the SNDP. Work on approximation algorithms for (ND) with any proper function started with a paper of Klein and Ravi [53], who gave a 3-approximation algorithm in the case that the proper function has range $\{0, 2\}$. Williamson, Goemans, Mihail, and Vazirani [71] gave the first approximation algorithm for general proper f; it has performance guarantee $2k$, where $k = \max_S f(S)$ (for SNDP $k = \max_{i,j} r_{ij}$). Goemans, Goldberg, Plotkin, Shmoys, Tardos, and Williamson [39] improved this to a $2H_k$-approximation algorithm, where $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

All of these algorithms use the primal-dual method in a sequence of k *phases*. To illustrate, we consider the algorithm of Goemans et al. Let F be the set of edges selected in phases 1 through $j-1$, and suppose we are now in phase j. In phase j, we form a hitting set problem in which we must hit all sets $T = \delta(S)$ with maximum *deficiency*, where the deficiency of a set S is $f(S) - |\delta(S) \cap F|$. We use the algorithm of Figure 3 to produce a set of edges A'. We add these to F, and start the next phase. Notice that each phase reduces the maximum deficiency by at least one, so that the maximum deficiency is at most $k - j + 1$ in phase j. Initially the deficiency is k, and when the deficiency of all sets is nonpositive, we have a feasible solution since F will contain at least $f(S)$ edges from $\delta(S)$ for each S. Roughly speaking, because the optimal solution to the SNDP has at least $k-j+1$ edges of $E-F$ hitting each set of maximum deficiency in phase j, the optimal solution to the hitting set problem in phase j costs at most $\frac{1}{k-j+1}$ times the optimum solution to SNDP. Goemans et al. prove that the primal-dual algorithm gives a hitting set solution of no more than twice the hitting set optimal. Thus the cost of the overall set of edges F the algorithm produces is at most $\sum_{j=1}^{k} \frac{2}{k-j+1}$ times the SNDP optimal, for a $2H_k$-approximation algorithm. Goemans et al. [39] show that their algorithm extends to the case of *weakly supermodular* functions f, a generalization of proper functions, when the minimally violated sets can be found in polynomial time (as they can for proper f); a function f is weakly supermodular if $f(V) = 0$ and for all $A, B \subseteq V$, either $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$ or $f(A) + f(B) \leq f(A - B) + f(B - A)$.

Mihail, Shallcross, Dean and Mostrel [58] implemented a variation of this algorithm for use in a telephone network design toolkit, and found that it works well in practice.

Recently, Jain [49] gave a non-primal-dual 2-approximation algorithm for (ND) for any weakly supermodular function f (assuming a certain polynomial-time separation oracle for f) by showing that any basic solution to the LP relaxation will always contain some $e \in E$ for which $x_e \geq 1/2$. The performance guarantee is obtained by rounding the value for this edge up to 1, then recursing on the remaining subproblem.

Although the performance guarantee of Jain's algorithm is much stronger than that given for the primal-dual algorithm above, the primal-dual algorithm is still of interest. Jain's algorithm requires solving the linear programming relaxation of (ND) with the integer constraints $x_e \in \{0,1\}$ replaced by $0 \leq x_e \leq 1$, which is a nontrivial computational task. The primal-dual algorithm is likely to be more efficient in practice.

3.2 Feedback vertex set problems

Another application of the primal-dual method has been to feedback vertex set problems, as we saw in Section 2. However, as we remarked in that section, the integrality gap for the hitting set formulation of the problem is at least $\Omega(\log n)$, and so we will not be able to obtain better performance guarantees unless we consider special cases of the problem, or different integer programming formulations. We consider both in turn in this section.

Goemans and Williamson [43] consider feedback vertex problems in planar graphs. They consider a class of hitting set problems, in which one must hit a select set of cycles $\mathcal{C}$ of a graph. Their class includes the feedback vertex set problem, the feedback vertex set problem in directed graphs (in which $\mathcal{C}$ is the set of directed cycles), the subset feedback vertex set problem (in which one is given a set of vertices S, and $\mathcal{C}$ is the set of cycles which contain some vertex of S), and the graph bipartization problem (in which $\mathcal{C}$ is the set of odd cycles; thus removing a solution set of vertices causes the remaining graph to be bipartite). They apply the algorithm of Figure 3 to these problems to obtain a $\frac{9}{4}$-approximation algorithm for these problems. To choose the collection of violated sets in the algorithm, they consider the *face-minimal* violated sets. Suppose each time the algorithm selects a vertex to add to A, we remove the vertex and incident edges from the graph, leaving a planar graph. Given a plane embedding, for any simple cycle $C \in \mathcal{C}$, let $F(C)$ be the set of faces of the graph interior to C. Then we say $C \in \mathcal{C}$ is face-minimal if there is no $C' \in \mathcal{C}$ such that $F(C') \subset F(C)$. The class of cycles $\mathcal{C}$ is such that in the collection of face-minimal cycles $\mathcal{F}$ $F(C) \cap F(C') = \emptyset$ for any two $C, C' \in \mathcal{F}$; the reader can verify this for the four problems given above. Goemans and Williamson [43] show that for any minimal solution D to these hitting set problems (that is, for any $v \in D$, $D - v$ does not hit all cycles in $\mathcal{C}$)

$$\sum_{C \in \mathcal{F}} |C \cap D| \leq 3|\mathcal{F}|,$$

giving a 3-approximation algorithm for these problems in planar graphs via Theorem 4. By carefully selecting a subset $\mathcal{F}' \subseteq \mathcal{F}$, they are able to replace the factor of 3 with $\frac{9}{4}$, leading to the claimed approximation algorithms.

Becker and Geiger [15] and Bafna, Berman, and Fujito [6] independently gave the first 2-approximation algorithms for the feedback vertex set problem in general undirected graphs. Because it will simplify our exposition somewhat, we will focus on the algorithm of Bafna et al. Their algorithm chooses vertices in a series of iterations, building up a feasible solution, and then removes excess vertices in a reverse delete step as in the algorithms of Figures 2 and 3. In a given iteration, the algorithm first checks to see whether the graph contains any *semi-disjoint* cycles. A semi-disjoint cycle is a cycle in which at most one vertex has degree greater than two. If the graph has a semi-disjoint cycle, the algorithm selects the cheapest vertex from the cycle. Otherwise, the algorithm selects the vertex that minimizes the ratio of weight to the vertex's degree minus one; that is, it chooses the $\arg\min_{v \in V} \frac{w_v}{d(v)-1}$, where $d(v)$ is the degree of the vertex. It then reduces the weight of every vertex in the graph by $\epsilon = \min_{v \in V} \frac{w_v}{d(v)-1}$. After the vertex is selected, it and all incident edges are removed from the graph, and all degree 1 vertices and incident edges are removed until none are left.

Chudak, Goemans, Hochbaum, and Williamson [20] have shown that this algorithm can be viewed as a primal-dual algorithm on a different integer programming formulation of the feedback vertex set problem. Here we give the formulation and show how a primal-dual algorithm is equivalent. To get the integer program, we first need the following lemma.

Lemma 6 *For any feedback vertex set F of a graph $G = (V, E)$,*

$$\sum_{v \in F} (d(v) - 1) \geq |E| - |V| + 1.$$

Proof: By the definition of a feedback vertex set, the removal of F leaves an acyclic graph. Therefore, once F and edges incident to F are removed from the graph, at most $|V| - |F| - 1$ edges remain. We remove at most $\sum_{v \in F} d(v)$ edges from the graph. Therefore, we have that $|E| \leq (|V| - |F| - 1) + \sum_{v \in F} d(v)$, and rearranging terms gives the statement of the lemma. $\Box$

Note that the lemma must still hold for the graph induced by any subset of vertices S; that is, if $G[S] = (V, E[S])$ is the graph induced by the subset of vertices S, and $d_S(v)$ is the degree of vertex v in $G[S]$, then for any feedback vertex set F, we have that $\sum_{v \in F} (d_S(v) - 1) \geq |E[S]| - |S| + 1$. We can use this to get the following integer programming formulation of the problem:

$$(FVS) \qquad \begin{array}{lll} \text{Min} & \displaystyle\sum_{v \in V} w_v x_v & \\ \text{subject to:} & & \\ & \displaystyle\sum_{v \in S} (d_S(v) - 1) x_v \geq b(S) & S \subseteq V \\ & x_v \in \{0, 1\} & v \in V, \end{array}$$

where $b(S) = |E[S]| - |S| + 1$. By the reasoning above, any feedback vertex set gives a feasible solution to the integer program (FVS). We can also show that any feasible solution x to (FVS) must be a feedback vertex set. Suppose not, and suppose there is some cycle C such that $x_v = 0$ for all $v \in C$. Consider the constraint of the integer program corresponding to C. Since there are at least $|C|$ edges in $E[C]$ (since it contains a cycle), the right-hand side of the constraint is at least 1, while the left-hand side is 0, which contradicts the feasibility of x.

We now give a primal-dual algorithm for the problem based on this integer programming formulation. First, we give the dual of a linear programming relaxation of (FVS):

$$(FVS - D) \qquad \begin{array}{lll} \text{Max} & \displaystyle\sum_{S} b(S) y_S & \\ \text{subject to:} & & \\ & \displaystyle\sum_{S: v \in S} (d_S(v) - 1) y_S \leq w_v & v \in V \\ & y_S \geq 0 & S \subseteq V. \end{array}$$

We give the primal-dual algorithm in Figure 4. It follows precisely the same format as the algorithm of Figure 2; here we increase the dual variable corresponding to a semi-disjoint cycle (if one exists) or the dual variable corresponding to the vertex set of the remaining graph. When some dual constraint becomes tight, we add the corresponding vertex to our solution. It is not difficult to see

```
y ← 0
F ← ∅
l ← 0
V' ← V; E' ← E
While F is not feasible
    l ← l + 1
    Recursively remove degree one vertices and edges from V' and E'
    If (V', E') contains a semi-disjoint cycle C
        S ← C
    Else
        S ← V'
    Increase y_S until ∃v_l ∈ S : Σ_{T:v_l∈T} (d_T(v_l) − 1)y_T = c_{v_l}
    F ← F ∪ {v_l}
    Remove v_l from V' and attached edges from E'.
For j ← l downto 1
    If F − {v_j} is feasible then F ← F − {v_j}
F' ← F
Output F' (and y)
```

Figure 4: A primal-dual version of the Bafna-Berman-Fujito algorithm for the feedback vertex set problem.

that the vertex selected by the primal-dual algorithm is exactly the same as that selected by the Bafna et al. algorithm. The cost of the vertices returned is

$$\sum_{v\in F'} w_v = \sum_{v\in F'} \sum_{S:v\in S} (d_S(v)-1)y_S = \sum_S \sum_{v\in S\cap F'} (d_S(v)-1)y_S.$$

To obtain a performance guarantee of 2, we wish to show that

$$\sum_S \sum_{v\in S\cap F'} (d_S(v)-1)y_S \le 2\sum_S b(S)y_S,$$

since the right-hand side is twice the dual objective function. We can do this if we can show that for any S such that $y_S > 0$, $\sum_{v\in S\cap F'}(d_S(v)-1) \le 2b(S)$. By the properties of the reverse delete, it can be shown that $S \cap F'$ is a minimal feedback vertex set for $G[S]$. If S is a semidisjoint cycle, clearly the inequality holds since a minimal feedback vertex set for $G[S]$ consists of a single vertex v, $d_S(v) - 1 = 1$, and $b(S) = 1$. Now suppose S is the vertex set of a graph which contains no semidisjoint cycle. Then the performance guarantee of 2 for the algorithm is implied by the following lemma.

Lemma 7 (Bafna et al. [6], Chudak et al. [20]) *For any minimal feedback vertex set F of a graph $G = (V,E)$ which contains no semidisjoint cycles,*

$$\sum_{v\in F} (d(v)-1) \le 2b(V).$$

Fujito [35] has extended this work to a primal-dual algorithm for node-deletion problems for *hereditary* graph properties derived from matroids. A property is hereditary if for any graph G that has the property, every subgraph of G also has the property. The property is derived from a matroid if the edge subsets satisfying the property correspond to independent sets of some matroid. Fujito studies the problem of deleting a minimum-weight set of nodes so that the remaining graph satisfies such a property. Consider the property of having no cycles: certainly this is hereditary, and it derives from the graphic matroid. The feedback vertex set problem is the corresponding node deletion problem. Fujito shows that if r^d is the rank function of the dual matroid, the following is an integer programming formulation of the problem:

$$\begin{array}{lll} \text{Min} & \sum_{v \in V} w_v x_v & \\ \text{subject to:} & & \\ & \sum_{v \in S} r^d(\delta(v)) x_v \geq r^d(E[S]) & S \subseteq V \\ & x_v \in \{0,1\} & v \in V. \end{array}$$

Note that in the case of the graphic matroid, $r^d(E[S]) = |E[S]| - |S| + c(G[S])$, where $c(G[S])$ is the number of connected components of S, and $r^d(\delta(v))$ is the degree of v minus the number of blocks containing v. Thus Fujito's integer program is almost the same as (FVS). He also gives a primal-dual 2-approximation algorithm for the feedback vertex set problem (as well as some others), and it is somewhat simpler than the one above in that the algorithm does not need a separate case for semi-disjoint cycles.

3.3 Prize-collecting problems

In this section, we consider a variation of *prize-collecting* problems introduced by Balas [8]. We will focus on the *prize-collecting Steiner tree problem* (PCST). In this problem we are given an undirected graph $G = (V, E)$, nonnegative costs on edges c_e, a root vertex r, and nonnegative penalties π_i on the vertices i. The goal is to find a tree T which includes r such that the cost of the edges in T plus the cost of the penalties of vertices not in T is minimized. Johnson, Minkoff, and Phillips [52] study this problem in the context of deciding which customers to connect to a cable system, forgoing the profits of customers who are not connected. The objective function given above minimizes the total cost of the cables and total profit lost.

The problem can be modelled as a hitting set problem in the following way. We have two different types of ground elements: each edge e is a ground element, and each subset $X \subseteq V - r$ is also a ground element. The cost of each edge is c_e, and the cost of a subset X is $\pi(X) = \sum_{i \in X} \pi_i$. For all $S \subseteq V - r$, we must hit the set $\delta(S) \cup \{X : X \supseteq S\}$; that is, either we must select an edge of $\delta(S)$, or we must choose some subset X that is a superset of S. We can show that this models the prize-collecting Steiner tree problem. If we have a tree T that is a solution for the PCST spanning the vertices $V(T)$, we can get a solution to the hitting set problem of no greater cost by including all edges from T and the subset X of the vertices $V - V(T)$. Clearly this hits all sets $S \subseteq V - V(T)$, and all other subsets $S \subseteq V - r$ must include some vertices of T, and thus $\delta(S)$ is hit by some edge of T. Similarly, given a solution to the hitting set problem, we construct a solution to PCST of no greater cost by taking as our tree T any tree spanning the connected component containing r. The hitting set solution must contain some $X \supseteq V - V(T)$ to hit the set $S = V - V(T)$, and thus the cost of the hitting set solution includes the penalties on vertices not spanned by T.

Thus we can model the PCST by the following integer program:

$$\begin{array}{lll}
\text{Min} & \displaystyle\sum_{e\in E} c_e x_e + \sum_{X\subseteq V} \pi(X) z_X & \\
\text{subject to:} & & \\
& \displaystyle\sum_{e\in\delta(S)} x_e + \sum_{X:X\supseteq S} z_X \geq 1 & S \subseteq V - r \\
& x_e \in \{0,1\} & e \in E \\
& z_X \in \{0,1\} & X \subseteq V.
\end{array}$$

Taking the dual of the linear programming relaxation, we obtain:

$$\begin{array}{lll}
\text{Max} & \displaystyle\sum_{S\subseteq V-r} y_S & \\
\text{subject to:} & & \\
& \displaystyle\sum_{S:e\in\delta(S)} y_S \leq c_e & e \in E \\
& \displaystyle\sum_{S:S\subseteq X} y_S \leq \pi(X) & X \subseteq V \\
& y_S \geq 0 & S \subseteq V - r.
\end{array}$$

We can apply the algorithm of Figure 3 and the analysis of Theorem 4 in a more or less straightforward fashion to obtain a 2-approximation algorithm for the PCST. Notice that the two types of ground elements in the hitting set formulation lead to two different types of packing constraints in the dual, one on edges and one on subsets of vertices. Thus as we increase dual variables, either an edge constraint can become tight (in which case we add the edge to our current solution), or a subset constraint can become tight (in which case we add the subset to our current solution). The minimal violated sets chosen in the algorithm of Figure 3 are connected components C of the set of selected edges such that $r \notin C$ and such that the subset C itself has not been selected by the algorithm (since then the primal constraint corresponding to the set C is not violated). One can use the analysis of Theorem 4 to show the following:

Theorem 8 (Goemans and Williamson [41]) *The algorithm of Figure 3 returns a tree T, a set of unspanned vertices X, and a feasible dual solution y such that*

$$\sum_{e\in T} c_e + 2\sum_{i\in X} \pi_i \leq 2 \sum_{S\subseteq V-r} y_S.$$

Johnson, Minkoff, and Phillips [52] have implemented this algorithm and found that it is usually within 5% of optimal on the instances they examined.

The algorithm for PCST can be used as a subroutine to obtain a 2-approximation algorithm for the *prize-collecting traveling salesman problem*, in which we must find a tour containing r that minimizes the cost of the tour plus the sum of the penalties of the vertices not visited by the tour.

4 Recent developments of the primal-dual method

We now turn to recent applications of the primal-dual method that do not fit so easily in the framework developed in Section 2. Interestingly, these developments give performance guarantees on variations of some dual-ascent heuristics considered earlier in the literature by Erlenkotter [29] for the uncapacitated facility location problem and Wong [73] for the Steiner tree problem.

4.1 Uncapacitated facility location

In the *uncapacitated facility location problem* (UFL), we are given as input a finite set of locations V, a subset $F \subset V$ of facilities, a set of facility costs $f_i \geq 0$ for all $i \in F$, a set of clients $D = V - F$, and a set of assignment costs $c_{ij} \geq 0$ for assigning client $j \in D$ to facility $i \in F$. We assume that these assignment costs obey the triangle inequality, in the sense that for clients j, k and facilities h, i, $c_{hk} \leq c_{hj} + c_{ij} + c_{ik}$. The goal is to select a set of facilities to open and to assign clients to these open facilities so as to minimize the total cost of open facilities and cost of the assignment.

The following integer programming formulation of the problem is due to Balinski [9]. We let the indicator variable y_i denote whether facility i is open, and the indicator variable x_{ij} denote whether client j has been assigned to facility i. Then the following IP models UFL:

$$
(UFL) \qquad
\begin{array}{llll}
\text{Min} & \displaystyle\sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij} & & \\
\text{subject to:} & & & \\
& \displaystyle\sum_{i \in F} x_{ij} = 1 & j \in D & (7) \\
& x_{ij} \leq y_i & i \in F, j \in D & (8) \\
& y_i \in \{0,1\} & i \in F & \\
& x_{ij} \in \{0,1\} & i \in F, j \in D. &
\end{array}
$$

The constraints (7) guarantee that each client is assigned to some facility, and those in (8) guarantee that a client is only assigned to an open facility. If we drop the integrality constraints and take the dual, we obtain

$$
(UFLD) \qquad
\begin{array}{llll}
\text{Max} & \displaystyle\sum_{j \in D} v_j & & \\
\text{subject to:} & & & \\
& \displaystyle\sum_{j \in D} w_{ij} \leq f_i & i \in F & (9) \\
& v_j - w_{ij} \leq c_{ij} & i \in F, j \in D & (10) \\
& w_{ij} \geq 0 & i \in F, j \in D. &
\end{array}
$$

We cannot apply the algorithm of Figure 3 in the most straightforward way here. So far all of our integer programming formulations have been covering IPs whose associated duals have been packing LPs, which is not the case for UFL. However, Jain and Vazirani [50] show that it is possible to get a 3-approximation algorithm by modifying the primal-dual algorithm somewhat. They set all the w_{ij} and v_j variables of $(UFLD)$ to zero, then increase the variables v_j uniformly. If for some i, j $v_j \geq c_{ij}$, they also increase w_{ij} at the same rate to maintain the feasibility of the constraints (10). Eventually for some facility i a constraint (9) becomes tight; to maintain feasibility, they stop increasing the variables v_j such that $v_j \geq c_{ij}$ for that facility i. This process continues until it is not possible to increase any v_j. Now consider the graph G of edges (i, j) such that for x_{ij} the corresponding dual inequality is tight (that is, when $v_j = c_{ij} + w_{ij}$, which occurs whenever $v_j \geq c_{ij}$). Rather than creating an assignment of clients to facilities solely from edges in this graph (as the primal-dual method developed so far would do), Jain and Vazirani carefully choose a subset of facilities to open of those whose corresponding dual constraint (9) is tight. The subset they open is such that no open facility is within a path of length two of any other open facility in G, but such that every client is within a path of length three of an open facility. Then by using the triangle inequality they are able to show that assigning client j to the nearest open facility does not cost more than $v_j - w_{ij}$ if j is next to an open facility i, and no more than $3v_j$ otherwise. Thus they are able to prove the following theorem, which implies that the algorithm is a 3-approximation algorithm.

Theorem 9 (Jain and Vazirani [50]) *The algorithm finds a feasible solution $(\bar{x}, \bar{y})$ to the IP (UFL) such that*

$$\sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + 3\sum_{i \in F} f_i \bar{y}_i \leq 3\sum_{j \in D} v_j.$$

Proof sketch: Suppose we divide the clients D into two sets: D_1, the clients that are next to an open facility in G, and D_3, the clients that are not. Note by the argument above that any client in D_1 is next to only one open facility. Then for the clients in D_1,

$$\sum_{j \in D_1, i \in F} c_{ij}\bar{x}_{ij} = \sum_{j \in D_1}\left(v_j - \sum_{i \in F} w_{ij}\bar{x}_{ij}\right) = \sum_{j \in D_1} v_j - \sum_{i \in F} f_i \bar{y}_i. \tag{11}$$

The first equality follows since for clients j in D_1, the constraint corresponding to i, j such that $\bar{x}_{ij} = 1$ is tight, and $c_{ij} = v_j - w_{ij}$. The second equality follows since all neighbors j of an open facility i are in D_1 and assigned to i, we have that $f_i = \sum_{j \in D} w_{ij} = \sum_{j \in D} w_{ij}\bar{x}_{ij}$. Multiplying the left-hand side of (11) by 3 gives the inequality

$$\sum_{j \in D_1, i \in F} c_{ij}\bar{x}_{ij} \leq 3\left(\sum_{j \in D_1} v_j - \sum_{i \in F} f_i \bar{y}_i\right) \tag{12}$$

We know from above that $\sum_{j \in D_3, i \in F} c_{ij}\bar{x}_{ij} \leq 3\sum_{j \in D_3} v_j$. Adding this inequality to (12) and rearranging terms gives

$$\sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + 3\sum_{i \in F} f_i \bar{y}_i \leq 3\sum_{j \in D} v_j,$$

as desired. □

4.2 Lagrangean relaxation and the k-median problem

The technique of Lagrangean relaxation has long been used in combinatorial optimization; the central idea is roughly that given a difficult integer or linear program to solve, one can often reduce the IP/LP to an easier IP/LP by removing some complicating constraints, but adding penalties for their violation to the objective function. Recently Jain and Vazirani [50] applied this technique to give an approximation algorithm for the *k-median problem*. The k-median problem has the same input as UFL, except that there are no costs for facilities, but rather an upper bound k on the number of facilities that can be opened. The goal is to open at most k facilities so as to minimize the cost of assigning clients to facilities.

We can give an integer programming formulation for the k-median problem much like that for the uncapacitated facility location problem:

$$\text{Min} \quad \sum_{i \in F, j \in D} c_{ij}x_{ij}$$

subject to:

$$\sum_{i \in F} x_{ij} = 1 \qquad j \in D$$

$$
\begin{array}{lll}
(kM) & x_{ij} \le y_i & i \in F, j \in D \\
& \sum_{i \in F} y_i \le k & \qquad (13) \\
& y_i \in \{0,1\} & i \in F \\
& x_{ij} \in \{0,1\} & i \in F, j \in D.
\end{array}
$$

The additional constraint (13) guarantees that no more than k facilities will be chosen.

Notice that if we apply Lagrangean relaxation to the complicating constraint (13), we obtain the following:

$$
\begin{array}{lll}
& \text{Min} \quad \sum_{i \in F, j \in D} c_{ij} x_{ij} + \lambda\left(\sum_{i \in F} y_i - k\right) & \\
& \text{subject to:} & \\
& \sum_{i \in F} x_{ij} = 1 & j \in D \\
(kMR) & x_{ij} \le y_i & i \in F, j \in D \\
& y_i \in \{0,1\} & i \in F \\
& x_{ij} \in \{0,1\} & i \in F, j \in D.
\end{array}
$$

This is a relaxation of (kM) since any feasible solution for (kM) will also be feasible and will have no greater cost (assuming $\lambda \ge 0$). The IP (kMR) is identical to the UFL formulation in which every facility cost is λ, except for the constant term $-\lambda k$ in the objective function. If we relax the integrality conditions and take the dual we obtain

$$
\begin{array}{lll}
& \text{Max} \quad \sum_{j \in D} v_j - k\lambda & \\
& \text{subject to:} & \\
(kMRD) & \sum_{j \in D} w_{ij} \le \lambda & i \in F \\
& v_j - w_{ij} \le c_{ij} & i \in F, j \in D \\
& w_{ij} \ge 0 & i \in F, j \in D.
\end{array}
$$

Observe any dual solution (v, w) for the facility location dual LP $(UFLD)$ with facility costs λ is feasible for $(kMRD)$. Furthermore, since $(kMRD)$ is the dual of the linear programming relaxation of (kMR), and (kMR) is a relaxation of the k-median formulation (kM), the objective function value of any feasible solution to $(kMRD)$ gives a lower bound on the cost of an optimal solution to the k-median problem.

Jain and Vazirani [50] use their UFL algorithm and the similarities between (UFL) and the Lagrangean relaxation for (kM) to obtain an approximation algorithm for the k-median problem. They observe that for facility cost $\lambda = 0$ their UFL algorithm will open all facilities, and for $\lambda = n \max_{i,j} c_{ij}$ the algorithm will only open one facility. So they perform a binary search on the value of λ, running the UFL algorithm each time in hopes of obtaining a solution $(\bar{x}, \bar{y})$ for (UFL) with $\sum_{i \in F} \bar{y}_i = k$. Suppose that this occurs. Then by Theorem 9, we know that

$$
\begin{aligned}
\sum_{i \in F, j \in D} c_{ij} \bar{x}_{ij} &\le 3\left(\sum_{j \in D} v_j - \sum_{i \in F} \lambda \bar{y}_i\right) \\
&= 3\left(\sum_{j \in D} v_j - \lambda k\right)
\end{aligned}
$$

Since the constructed dual solution (v, w) is feasible for $(UFLD)$ with facility costs λ, (v, w) is feasible for $(kMRD)$, and thus $\sum_{j \in D} v_j - \lambda k$ is a lower bound on the value of the optimal k-median. Hence if we can find a value of λ such that the UFL algorithm opens exactly k facilities, the solution is within a factor of 3 of optimal for the k-median problem.

However, in general such a value of λ may not exist. In this case, Jain and Vazirani find two solutions for two values of λ sufficiently close, one which opens more than k facilities, and one which opens fewer than k facilities. They then show that an appropriate convex combination of the inequalities of Theorem 9 for the two solutions gives an inequality showing that the convex combination of the primal solutions is no more than 3 times the value of a feasible dual solution for $(kMRD)$. Jain and Vazirani then show that by using the two solutions they can find a solution to the k-median problem that costs no more than twice the cost of the convex combination of the two solutions. In this way they obtain a 6-approximation algorithm for the k-median problem. Charikar and Guha [19] are able to improve this algorithm to a 4-approximation algorithm by carefully considering the differences in solutions produced by the algorithm for UFL for values of λ sufficiently close together.

Garg [37] implicitly used the technique above for solving the problem of finding a minimum tree spanning k vertices. In this problem, given an undirected graph with nonnegative costs c_e on the edges, a root vertex $r \in V$, and a positive integer k, one must find a minimum-cost tree including r that spans at least k vertices. Garg uses the prize-collecting Steiner tree algorithm mentioned in Section 3.3 as subroutine while doing a binary search on a parameter λ. Chudak, Roughgarden, and Williamson [21] make explicit Garg's use of Lagrangean relaxation. Consider the following integer program which models the problem of finding a minimum tree spanning k vertices:

$$
\begin{aligned}
\text{Min} \quad & \sum_{e \in E} c_e x_e \\
\text{subject to:} \quad & \\
& \sum_{e \in \delta(S)} x_e + \sum_{X : X \supseteq S} z_X \geq 1 && S \subseteq V - r \\
& \sum_{X \subseteq V} |X| z_X \leq |V| - k && (14) \\
& x_e \in \{0, 1\} && e \in E \\
& z_X \in \{0, 1\} && X \subseteq V.
\end{aligned}
$$

By using Lagrangean relaxation on the complicating constraint (14), we get an integer program of the same form as that for the prize-collecting Steiner tree, in which each penalty π_i is the Lagrangean variable λ. Garg [37] gives a simple 5-approximation for the problem, and a more complicated 3-approximation algorithm that depends on understanding the changes in the solution generated by the prize-collecting Steiner tree algorithm for small perturbations of λ. Chudak et al. [21] show that these proofs can be made to follow an outline similar to that for the k-median problem above.

4.3 The Steiner tree problem

In Section 3.1, we described a primal-dual 2-approximation algorithm for the Steiner tree problem. One unsatisfying aspect of this algorithm is that the integer programming formulation (ND) with the appropriate function f for the Steiner tree problem has an integrality gap of two, even in the case when the set of terminals $T = V$; that is, when the problem is a minimum spanning tree problem. In this case there are exact integer programming formulations. Pick an arbitrary root vertex r, and let $G' = (V, A)$ be a directed graph formed from the undirected graph G by replacing each undirected edge $e = (i, j)$ of cost c_e with two oppositely oriented arcs $a = (i, j)$ and $a' = (j, i)$, both of cost $c_a = c_{a'} = c_e$. Then the following linear program models the minimum spanning tree problem:

$$\begin{array}{lll} \text{Min} & \displaystyle\sum_{a \in A} c_a x_a & \\ \text{subject to:} & & \\ & \displaystyle\sum_{a \in \delta^-(S)} x_a \geq 1 & S \subseteq V - r \\ & x_a \geq 0 & a \in A. \end{array}$$

Recall that $\delta^-(S)$ is the set of arcs in a digraph with their heads in $S \subseteq V$, and tails not in S. The proof of correctness of Edmonds' branching algorithm given in Section 2 shows that this is an exact formulation.

We can modify the formulation to give a integer programming formulation of the Steiner tree problem. Let T be the set of terminals to be joined, and let r be an arbitrary member of T. Then the following integer program models the Steiner tree problem:

$$\begin{array}{lll} \text{Min} & \displaystyle\sum_{a \in A} c_a x_a & \\ \text{subject to:} & & \\ & \displaystyle\sum_{a \in \delta^-(S)} x_a \geq 1 & S \subseteq V - r, S \cap T \neq \emptyset \\ & x_a \in \{0, 1\} & a \in A. \end{array}$$

This is sometimes called the *bidirected* formulation of the Steiner tree problem. Notice that this formulation corresponds to a hitting set problem in which we must hit every set $\delta^-(S)$ for which $S \subseteq V - r, S \cap T \neq \emptyset$. The dual of the linear programming relaxation is

$$\begin{array}{lll} \text{Max} & \displaystyle\sum_{S: S \subseteq V - r, S \cap T \neq \emptyset} y_S & \\ \text{subject to:} & & \\ & \displaystyle\sum_{S: a \in \delta^-(S)} y_S \leq c_a & a \in A \\ & y_S \geq 0 & S \subseteq V - r, S \cap T \neq \emptyset. \end{array}$$

We could apply the algorithm of Figure 3 in some fashion to the problem, but it has not been clear how to obtain a good performance guarantee via Theorem 4. Heuristically some variant of the algorithm seems to give good results, however; Wong [73] shows that choosing any single violated set (as in the algorithm of Figure 2) gives solutions within 1% of optimal on small random instances.

Rajagopalan and Vazirani [63] give a modification to the primal-dual method that gives a $(\frac{3}{2} + \epsilon)$-approximation algorithm for the Steiner tree problem on *quasi-bipartite* graphs. They call a graph quasi-bipartite if every edge has at least one endpoint that is a terminal. Their algorithm combines a local search algorithm with a primal-dual algorithm. Given the set T of terminals and some subset X of non-terminals, they use a variation of the algorithm of Figure 3 in which they increase the duals of all minimal sets S that contain some but not all vertices of $T \cup X$, and such that $\delta^-(S)$ contains no edge of the current solution. Observe that this algorithm may increase dual variables y_S that do not contribute to the dual objective function, either because $T \cap S = \emptyset$

or $r \in S$. If the primal solution obtained ends up using a vertex v not in $T \cup X$, they show that adding v to X can only improve the cost of the resulting solution, so v is added to X and they iterate. If no such v is added, then they show that the cost of their solution F equals the value of the sum of the dual variables; that is, $\sum_{a \in F} c_a = \sum_S y_S$. Rajagopalan and Vazirani show that the total value of duals that do not contribute to the objective is no more than $\frac{1}{3} \sum_S y_S$, so that $\sum_{a \in F} c_a \leq \frac{3}{2} \sum_{S: S \subseteq V-r, S \cap T \neq \emptyset} y_S$, and therefore the cost of F is no more than $\frac{3}{2}$ times the optimal value. Măndoiu, Vazirani, and Ganley [59] give experimental results with this algorithm.

5 Conclusion and Open Questions

Even in this lengthy survey, it has not been possible to be comprehensive. For example, Bar-Noy, Bar-Yehuda, Freund, Naor, and Schieber [10] give an application of the primal-dual method to scheduling problems, Garg, Vazirani, and Yannakakis [38] to cut problems in trees, and Bertsimas and Teo [18] to several different problems. Interestingly, the paper of Bar-Noy et al. gives the first primal-dual approximation algorithm for a natural maximization problem; all previous applications have been to minimization problems.

In addition, we have not been able to describe the connection of the primal-dual method to the *local-ratio theorem* [13, 6, 11]. The two methods appear to be strongly related, though as of the writing of this survey no formal connection has been shown. In some cases, an approximation algorithm has been designed first using the local-ratio theorem, and then shown to have a primal-dual approximation algorithm; for example, this was the case for the feedback vertex set problem described in Section 3.2 [6, 20], and the scheduling problem of Bar-Noy et al. [10]. In the first case, the integer program modelling the problem had to be inferred from the design of the local-ratio algorithm.

We close this survey by listing several open problems of interest in this area.

1. The algorithm of Jain [49] shows that the integrality gap is 2 for the network design formulation (ND) of Section 3.1 for any weakly supermodular function f. Thus it is possible that there is also a primal-dual algorithm for any weakly supermodular function that has a performance guarantee of 2. Such an algorithm would be very interesting, and possibly more practical than Jain's. It would even be interesting to provide a primal-dual 2-approximation algorithm for the survivable network design problem in which one is allowed to have multiple copies of edges (that is, $x_e \in N$ rather than $x_e \in \{0, 1\}$).

2. It would be interesting to have a primal-dual approximation algorithm for the Steiner tree problem with performance guarantee better than 2 by using the bidirected formulation. On the other hand, perhaps no such performance guarantee is possible because the integrality gap of the formulation is at least $2 - \epsilon$ for any $\epsilon > 0$. A proof of this fact would also be of interest.

3. The primal-dual method for approximation algorithms shown in this survey are essentially dual ascent algorithms. The standard primal-dual method for combinatorial optimization problems can sometimes result in very complicated dual adjustment schemes (for example, Edmonds' blossom algorithm for weighted non-bipartite matching [24]). Can a more complicated scheme result in new or improved approximation algorithms for NP-hard problems in combinatorial optimization?

Acknowledgements

The author would like to thank Tim Roughgarden, David Shmoys, Madhu Sudan, and the two anonymous referees for several comments that improved the presentation of this survey.

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*, 24:440–456, 1995.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45:501–555, 1998.

[4] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45:70–122, 1998.

[6] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12:289–297, 1999.

[7] A. Balakrishnan, T. L. Magnanti, and R. Wong. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37:716–740, 1989.

[8] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[9] M. L. Balinski. Integer programming: methods, uses, computation. *Management Science*, 12:253–313, 1965.

[10] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[11] R. Bar-Yehuda. One for the price of two: a unified approach for approximating covering problems. *Algorithmica*, to appear. See also the proceedings of APPROX '98.

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

[13] 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.

[14] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27:942–959, 1998.

[15] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83:167–188, 1996.

[16] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability – towards tight results. *SIAM Journal on Computing*, 27:804–915, 1998.

[17] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 294–304, 1993.

[18] D. Bertsimas and C.-P. Teo. From valid inequalities to heuristics: A unified view of primal-dual approximation algorithms in covering problems. *Operations Research*, 46:503–514, 1998.

[19] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[20] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22:111–118, 1998.

[21] F. A. Chudak, T. Roughgarden, and D. P. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. Manuscript, 2000.

[22] 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, Princeton, NJ, 1956.

[23] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[24] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[25] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71B:233–240, 1967.

[26] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5:88–124, 1973.

[27] P. Erdős. Gráfok páros körüljárású részgráfjairól (On bipartite subgraphs of graphs, in Hungarian). *Mat. Lapok*, 18:283–288, 1967.

[28] P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math Debrecen*, 9:3–12, 1962.

[29] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

[30] G. Even, J. S. Naor, B. Schieber, and L. Zosin. Approximating minimum subset feedback sets in undirected graphs with applications. *SIAM Journal on Discrete Mathematics*, 13:255–267, 2000.

[31] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[32] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43:268–292, 1996.

[33] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[34] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[35] T. Fujito. Approximating node-deletion problems for matroidal properties. *Journal of Algorithms*, 31:211–227, 1999.

[36] H. N. Gabow, M. X. Goemans, and D. P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82:13–40, 1998.

[37] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 302–309, 1996.

[38] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[39] M. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[40] M. X. Goemans and D. P. Williamson. Approximating minimum-cost graph problems with spanning tree edges. *Operations Research Letters*, 16:183–189, 1994.

[41] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[42] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1997.

[43] M. X. Goemans and D. P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18:37–59, 1998.

[44] R. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[45] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[46] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182:105–142, 1999.

[47] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.

[48] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1997.

[49] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1998.

[50] K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[51] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[52] D. S. Johnson, M. Minkoff, and S. Phillips. The prize-collecting Steiner tree problem: theory and practice. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 760–769, 2000.

[53] P. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 39–55, 1993. Also appears as Brown University Technical Report CS-92-30.

[54] P. N. Klein. A data structure for bicategories, with application to speeding up an approximation algorithm. *Information Processing Letters*, 52:303–307, 1994.

[55] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[56] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[57] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994.

[58] M. Mihail, D. Shallcross, N. Dean, and M. Mostrel. A commercial application of survivable network design: ITP/INPLANS CCS network topology analyzer. To appear in the *Proceedings of the Seventh Annual Symposium on Discrete Algorithms*, 1996.

[59] I. I. Măndoiu, V. V. Vazirani, and J. L. Ganley. A new heuristic for rectilinear Steiner trees. In *Proceedings of the IEEE-ACM International Conference on Computer Aided Design*, 1999.

[60] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[61] C. H. Papadimitriou and S. Vempala. On the approximability of the traveling salesman problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[62] S. Raghavan. *Formulations and algorithms for network design problems with connectivity requirements.* PhD thesis, MIT, 1994.

[63] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999.

[64] H. Saran, V. Vazirani, and N. Young. A primal-dual approach to approximation algorithms for network Steiner problems. In *Proceedings of Indo-US workshop on Cooperative Research in Computer Science*, pages 166–168, 1992.

[65] D. B. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. In W. Cook, L. Lovász, and P. D. Seymour, editors, *Combinatorial Optimization*, pages 355–397. American Mathematical Society, 1995.

[66] C.-P. Teo. *Constructing approximation algorithms via linear programming relaxations: primal dual and randomized rounding techniques.* PhD thesis, MIT, 1996.

[67] V. V. Vazirani. *Approximation algorithms.* Springer, 2000.

[68] V. G. Vizing. On an estimate of the chromatic class of a p-graph (in Russian). *Diskret. Analiz.*, 3:23–30, 1964.

[69] D. P. Williamson. *On the design of approximation algorithms for a class of graph problems.* PhD thesis, MIT, Cambridge, MA, September 1993. Also appears as Tech Report MIT/LCS/TR-584.

[70] D. P. Williamson and M. X. Goemans. Computational experience with an approximation algorithm on large-scale Euclidean matching instances. *INFORMS Journal on Computing*, 8:29–40, 1996.

[71] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. An approximation algorithm for general graph connectivity problems. *Combinatorica*, 15:435–454, 1995.

[72] P. Winter. Steiner problem in networks: a survey. *Networks*, 17:129–167, 1987.

[73] R. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.