Dual Simplex

Mihai Banciu
School of Management
Bucknell University
Lewisburg, PA 17837
mmb018@bucknell.edu

January 2010

Abstract

The dual simplex algorithm is an attractive alternative as a solution method for linear programming problems. Since the addition of new constraints to a problem typically breaks primal feasibility but not dual feasibility, the dual simplex can be deployed for rapid reoptimization, without the need of finding new primal basic feasible solutions. This is especially useful in integer programming, where the use of cutting plane techniques require the introduction of new constraints at various stages of the branch-and-bound/cut/price algorithms. In this article, we give a detailed synopsis of the dual simplex method, including its history and its relationship to the primal simplex algorithm, as well as its properties, implementation challenges, and applications.

Consider the following linear programming problem $\mathbf{P}$ expressed in standard form:

$$
\begin{aligned}
[\mathbf{P}] \qquad & \min \mathbf{c}^T\mathbf{x} \\
\text{subject to:} & \\
& \mathbf{A}\mathbf{x} = \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

where $\mathbf{c}^T = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n$ is an n-dimensional *cost vector*, $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m\times n}$ is a matrix of *constraint coefficients*, $\mathbf{b} = (b_1, b_2, \ldots, b_m)^T \in \mathbb{R}^m$ is an m-dimensional *right hand side vector*, and $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is an n-dimensional vector of *decision variables*. We refer to the total cost $z = \mathbf{c}^T\mathbf{x} : \mathbb{R}^n \mapsto \mathbb{R}$ as the *objective value* of the problem $\mathbf{P}$. Stated in this form, $\mathbf{P}$ is called the *primal* problem, and the vector $\mathbf{x}^*$ that minimizes the objective function $\mathbf{c}^T\mathbf{x}^*$, while satisfying the equality condition $\mathbf{A}\mathbf{x}^* = \mathbf{b}$, is called the *optimal solution* to the problem $\mathbf{P}$. Associated with the primal problem $\mathbf{P}$, there exists a *dual* problem $\mathbf{D}$, which is formulated in the following way:

$$
\begin{aligned}
[\mathbf{D}] \qquad & \max \mathbf{b}^T\mathbf{u} \\
\text{subject to:} & \\
& \mathbf{A}^T\mathbf{u} \leq \mathbf{c} \\
& \mathbf{u} \text{ free}
\end{aligned}
$$

Here, the vector $\mathbf{u} = (u_1, u_2, \ldots, u_m) \in \mathbb{R}^m$ is called the vector of *dual variables*. Notice that formulating the dual problem of $\mathbf{D}$ will yield the original primal problem $\mathbf{P}$. The relationship between problems $\mathbf{P}$ and $\mathbf{D}$ is established by the following two theorems (the proofs of these and any subsequent result presented in this encyclopedic entry can be found in any introductory linear programming textbooks-see for example [1] or [2], as well as section 1.1.1.8 from the encyclopedia):

Theorem 1 (**Weak Duality**)**.** *If* $\mathbf{x}$ *is a feasible solution of* $\mathbf{P}$*, and* $\mathbf{u}$ *is a feasible solution to* $\mathbf{D}$*, then* $\mathbf{b}^T\mathbf{u} \leq \mathbf{c}^T\mathbf{x}$.

Theorem 2 (**Strong Duality**)**.** *If problem* $\mathbf{P}$ *has an optimal solution, then problem* $\mathbf{D}$ *also has an optimal solution. Moreover, the respective optimal solution values are equal to one another.*

An immediate consequence of weak duality is that if the primal problem $\mathbf{P}$ is unbounded from below, then its dual $\mathbf{D}$ must be infeasible, and correspondingly, if the dual problem $\mathbf{D}$ is unbounded from above, then the primal $\mathbf{P}$ must be infeasible. Additionally, if the choice of $\mathbf{x}$ and $\mathbf{u}$ is such that the inequality described in the weak duality theorem becomes an equality, then $\mathbf{x}$ and $\mathbf{u}$ are the optimal solutions to $\mathbf{P}$ and $\mathbf{D}$, respectively. Optimality of $\mathbf{x}$ and $\mathbf{u}$ can also be verified by way of the complementary slackness conditions:

Theorem 3 (**Complementary Slackness**)**.** *Let* $\mathbf{x}$ *and* $\mathbf{u}$ *be feasible solutions to* $\mathbf{P}$ *and* $\mathbf{D}$*, respectively. Then,* $\mathbf{x}$ *and* $\mathbf{u}$ *are optimal solutions for each problem if and only if* $\mathbf{u}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = 0$ *and* $(\mathbf{c} - \mathbf{A}^T\mathbf{u})^T\mathbf{x} = 0$.

Theorem 3 implies three things. First, observe that in problem $\mathbf{P}$, the first complementary slackness condition, i.e., $\mathbf{u}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = 0$, is automatically satisfied (since $\mathbf{P}$ is expressed in standard form). Second, the complementary slackness theorem states that in problem $\mathbf{D}$, if any constraint is not tight, then the corresponding value of the primal variable corresponding to each such constraint is 0. If the constraint is tight, then the corresponding primal value is non-negative. Finally, the theorem provides an easy way of translating primal solutions into corresponding dual solutions, and vice-versa, as well as providing a certificate for whether a proposed solution for $\mathbf{P}$ is, in fact, optimal. The following example illustrates this last property.

Example (**Complementary Slackness**)**.** Consider the following LP expressed in standard form, together with its dual:

$$
\begin{aligned}
\max\ & 3x_1 + 4x_2 - x_3 \\
\text{subject to: } & 3x_1 - 4x_2 + 2x_3 = 8 \\
& 4x_1 - 2x_2 + 5x_3 = 22 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
\min\ & 8u_1 + 22u_2 \\
\text{subject to: } & 3u_1 + 4u_2 \leq 3 \\
& -4u_1 - 2u_2 \leq 4 \\
& 2u_1 + 5u_2 \leq -1
\end{aligned}
$$

Suppose we are presented with the solution $\mathbf{x} = (0, 0.25, 4.5)$ and wish to establish optimality via complementary slackness. The first complementary slackness condition is automatically satisfied because the problem is in standard form. Since $x_2 > 0$ and $x_3 > 0$, from the second complementary slackness condition we obtain $-4u_1 - 2u_2 = 4$ and $2u_1 + 5u_2 = -1$, which solves to $\mathbf{u} = (-1.125, 0.25)$. This solution is dual feasible, as it also satisfies the first constraint of the dual problem, and the total cost is -3.5. This is the same cost of the primal problem; therefore, by the strong duality theorem, $\mathbf{x}$ is an optimal solution to the original LP.

The following theorem establishes the Karush–Kuhn–Tucker conditions for a linear programming problem. For more advanced discussions on the topic, consult section 1.1.1.8 of the encyclopedia.

Theorem 4 (KKT optimality conditions for LP). *Given a linear programming problem* $\mathbf{P}$*, the vector* $\mathbf{x}$ *is an optimal solution to* $\mathbf{P}$ *if and only if there exist vectors* $\mathbf{u}$ *and* $\mathbf{r}$ *such that:*

1. $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ *(primal feasibility)*
2. $\mathbf{A}^T\mathbf{u} + \mathbf{r} = \mathbf{c}, \mathbf{r} \geq \mathbf{0}$ *(dual feasibility)*
3. $\mathbf{r}^T\mathbf{x} = 0$ *(complementary slackness)*

In Theorem 4, the vector $\mathbf{r}$ is simply the vector of slack/surpluses $\mathbf{c} - \mathbf{A}^T\mathbf{u}$ from the dual problem $\mathbf{D}$. In the primal problem $\mathbf{P}$, the vector $\mathbf{r}$ is referred to as the vector of *reduced costs*, and it is usually denoted by $\bar{\mathbf{c}}$.

From a computational perspective, the importance of the theorem is that a computer algorithm designed to solve problem $\mathbf{P}$ using an extreme vertex approach on the polyhedron described by the constraints of $\mathbf{P}$, can work in one of two possible ways. The first approach is to make sure that both primal feasibility and complementary slackness are preserved at every iteration of the algorithm, while seeking dual feasibility only at the optimal solution. The second approach is to ensure dual feasibility and complementary slackness during every iteration, while seeking primal feasibility only at the optimal solution. The first approach yielded George Dantzig's (primal) simplex algorithm around 1947 [3, 4, 5], which is, arguably, the cornerstone of the entire linear optimization literature. The second method led to the discovery of the dual simplex algorithm, in 1954 by Carlton Lemke [6].

While the simplex algorithm, in its both incarnations, has shown remarkable good performance in practice, in that it usually requires $\mathcal{O}(m)$ iterations (pivots), there exist instances where an exponential number of iterations may be required until an optimal solution is found (see [7] for the first description of instances that lead to a performance on the order of $\mathcal{O}(2^n)$). Hence, later theoretical efforts concentrated on a different attack for solving the linear programming problem. These efforts were successful, first in 1979 when Leonid Khachian [8] discovered the ellipsoidal method for solving linear programming problems in polynomial time, and later in 1984 with Narendra Karmarkar's discovery of the first interior point method [9], which created a very fertile research area in subsequent years (see section 1.1.3 of the encyclopedia for more information). In the following section we will discuss the theory and the implementation of the dual simplex algorithm.

1 The Dual Simplex Algorithm

Throughout this section, we assume familiarity with basic linear algebra concepts, such as bases, spaces, and polyhedra, as well as an understanding of the (revised) simplex algorithm, which is described in section 1.1.1.3 of this encyclopedia.

Assume that the coefficient matrix $\mathbf{A}$ is of full rank m $(m < n)$, and that there exist a basis $\mathbf{B}$ corresponding to the m linearly independent columns of $\mathbf{A}$, and a matrix $\mathbf{N}$ that contains the remaining $n - m$ non-basic columns of $\mathbf{A}$. We partition the solution vector $\mathbf{x}$ into two components $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} = (x_1, x_2, \ldots, x_m)^T$ and $\mathbf{x}_N = (x_{m+1}, x_{m+2}, \ldots, x_n)^T = \mathbf{0}$, called the *basic* and *non-basic* solutions, respectively. In a similar fashion, we partition the cost vector $\mathbf{c}^T = [\mathbf{c}_B|\mathbf{c}_N]$ into a basic and a non-basic component, that is, $\mathbf{c}_B$ and $\mathbf{c}_N$ are the cost coefficients associated with $\mathbf{x}_B$ and $\mathbf{x}_N$, respectively. Suppose, moreover, that the basis $\mathbf{B}$ induces a dual feasible solution $\mathbf{u}$, that is, $\mathbf{A}^T\mathbf{u} \leq \mathbf{c}$ and $\mathbf{u}^T = \mathbf{c}_B^T\mathbf{B}^{-1}$. It is easy to see that the equality constraints of $\mathbf{P}$ are satisfied, since

$$\mathbf{A}\mathbf{x} = [\mathbf{B}|\mathbf{N}]\left[\frac{\mathbf{B}^{-1}\mathbf{b}}{\mathbf{0}}\right] = \mathbf{b}$$

and that complementary slackness is also satisfied, since

$$(\mathbf{c} - \mathbf{A}^T\mathbf{u})^T\mathbf{x} = \mathbf{c}^T\mathbf{x} - \mathbf{u}^T\mathbf{A}\mathbf{x} = \mathbf{c}^T\mathbf{B}^{-1}\mathbf{b} - \mathbf{c}^T\mathbf{B}^{-1}\mathbf{b} = 0$$

The only ingredient missing for an optimal solution to $\mathbf{P}$ is the requirement that $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$. If this inequality happens to hold, then $\mathbf{x}$ is optimal for $\mathbf{P}$, since all conditions of Theorem 4 are met. On the other hand, if there exists at least one negative entry in $\mathbf{x}_B$, then we perform a change of basis that eliminates this entry from $\mathbf{B}$, thus reducing the primal infeasibility. The procedure repeats until all entries in $\mathbf{x}_B$ are non-negative. This is the essence of the dual simplex algorithm.

Naturally, the two questions that arise are about how this change of basis is actually performed, and whether or not the new solution obtained after the basis change is actually an improvement over the old one. The following sub-sections describe the actual flow of the dual simplex algorithm, alongside with a discussion on several theoretical and practical implementation challenges.

1.1 Summary of the dual simplex method: description and geometry

The dual simplex method entails the following steps.

Step 0 (Initialization) Find a dual feasible basis $\mathbf{B}$ such that the associated reduced cost vector $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{u}^T\mathbf{A}$ is non-negative (in the sense that each vector entry is non-negative). (We will later provide a quick procedure for identifying such a basis.)

Step 1 (Pricing) If $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, stop; the current solution is optimal. Otherwise, select a pivot row i with the corresponding value of the basic variable $x_i < 0$. If there are multiple candidates, select the one with the minimum value. The variable x_i will leave the basis.

Step 2 (Ratio test) If all constraint coefficients a_{ij} alongside the i^{th} row are nonnegative ($a_{ij} \geq 0, \forall j$), stop. The dual problem is unbounded, and therefore the primal problem is infeasible. Otherwise, select the pivot column j, by performing the following minimum ratio test:

$$j = \arg\min_{k=1,..,m} \left\{ \frac{\bar{c}_k}{a_{ik}} \mid a_{ik} < 0 \right\}$$

The corresponding variable x_j will enter the basis.

Step 3 (Basis change) Pivot on element a_{ij} and go to step 1.

We will illustrate the dual simplex method by contrasting it with the primal simplex, on a simple example, presented below. We will use the traditional simplex tableau exposition, presented in the following fashion (see [10] for one of the first descriptions of the simplex tableau, and [11] for a detailed description of this particular implementation form):

Table 1: The general form of a simplex tableau

	z	$\mathbf{x}_B$	$\mathbf{x}_N$	RHS
z	1	0	$\mathbf{c}_B\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_N$	$\mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b}$
$\mathbf{x}_B$	0	$\mathbf{I}$	$\mathbf{B}^{-1}\mathbf{N}$	$\mathbf{B}^{-1}\mathbf{b}$

Note that in other treatments of linear programming (e.g. [1]), the reduced costs are defined the other way around, that is, $\mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}$. Under this interpretation, the arguments presented below still hold, but the signs are reversed—for example, the algorithm would terminate when all reduced costs are non–negative (in a minimization example), rather than non-positive, as we assume.

Example (Primal Simplex). Consider the following linear programming problem, together with its corresponding dual:

$$\begin{aligned} &\min 2x_1 + x_2 \\ &\text{subject to: } x_1 + x_2 \geq 3 \\ &\qquad x_1 \geq 2 \\ &\qquad x_1, x_2 \geq 0 \end{aligned} \qquad\qquad \begin{aligned} &\max 3u_1 + 2u_2 \\ &\text{subject to: } u_1 + u_2 \leq 2 \\ &\qquad u_1 \leq 1 \\ &\qquad u_1, u_2 \geq 0 \end{aligned}$$

The following primal simplex tableaus solve the primal and the dual problem. For the primal problem, we use the big-M method to form an initial basis, rather than the 2-phase method, to keep everything in one tableau.

The optimal solution to the primal problem is $\mathbf{x} = (2, 1, 0, 0, 0, 0)$; the corresponding optimal dual vector is $\mathbf{u} = (1, 1, 0, 0)$. The optimal value of the objective function is $z = 5$. One can also quickly check the complementary slackness conditions and verify that they are met.

We will examine now the way in which the dual simplex algorithm operates. First, we look at the primal tableau, with the starting basis given by the surplus variables x_3 and x_4

Table 2a: Primal simplex on primal problem

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	$2M-2$	$M-1$	$-M$	$-M$	0	0	$5M$
x_5	0	1	1	-1	0	1	0	3
x_6	0	1^*	0	0	-1	0	1	2
z	1	0	$M-1$	$-M$	$M-2$	0	$-2M+2$	$M+4$
x_5	0	0	1^*	-1	1	1	-1	1
x_1	0	1	0	0	-1	0	1	2
z	1	0	0	-1	-1	$-M+1$	$-M+1$	5
x_2	0	0	1	-1	1	1	-1	1
x_1	0	1	0	0	-1	0	1	2

Table 2b: Primal simplex on dual problem

	z	u_1	u_2	u_3	u_4	RHS
z	1	3	2	0	0	0
u_3	0	1	1	1	0	2
u_4	0	1^*	0	0	1	1
z	1	0	2	0	-3	-3
u_3	0	1^*	1	1	0	2
u_1	0	1	0	0	1	1
z	1	0	0	-2	-1	-5
u_2	0	0	1	1	-1	1
u_1	0	1	0	0	1	1

Table 3a: Initial basis for dual simplex

	z	x_1	x_2	x_3	x_4	RHS
z	1	-2	-1	0	0	0
x_3	0	-1	-1^*	1	0	-3
x_4	0	-1	0	0	1	-2

(and thus we multiply both rows by -1 to obtain the basis given by the identity matrix $\mathbf{I}_2$). Notice that this basis is primal infeasible, because of the negative terms in $\mathbf{B}^{-1}\mathbf{b}$.

We are at step 1 of the dual simplex method. We select x_3 as the leaving variable ($-3 < -2$). According to step 2, the problem is not infeasible (two negative entries on the selected row). The entering variable is x_2, by the minimum ratio test: $\min\{\frac{-2}{-1}, \frac{-1}{-1}\} = 1$. The new tableau is obtained by subtracting row 1 from row 0, and by dividing row 1 to -1:

Table 3b: First pivot using dual simplex

	z	x_1	x_2	x_3	x_4	RHS
z	1	-1	0	-1	0	3
x_2	0	1	1	-1	0	3
x_4	0	-1^*	0	0	1	-2

The solution is still primal infeasible. The variable x_4 leaves the basis, and x_1 enters, by the minimum ratio test. Row 2 gets subtracted from row 0, row 2 is added to row 1, and row 2 is divided by -1 to get the new tableau:

Table 3c: Second pivot using dual simplex

	z	x_1	x_2	x_3	x_4	RHS
z	1	0	0	-1	-1	5
x_2	0	0	1	-1	1	1
x_1	0	1	0	0	-1	2

Since all terms in $\mathbf{B}^{-1}\mathbf{b}$ are positive and all entries in row 0 are negative, this is the optimal basis. The algorithm terminates, with the same solution as reported by the primal simplex above. Since the dual problem maximizes the objective function, the increase in z from one pivot to the next is natural.

Several observations are in order. First, notice the correspondence between the dual simplex, as exhibited in Tables 3a through 3c, and the primal simplex method applied to the dual problem, as shown in Table 2b. Through simple transposition manipulations, we can see that the two tableaus are equivalent. This is important: from a mathematical point of view, the dual simplex method used to solve a primal problem is equivalent to the primal simplex method applied to the dual problem, as long as the dual is not converted into standard form. (Converting the dual problem into standard form could lead to different tableaus.) Second, from a geometrical perspective, the dual simplex iterates around infeasible primal solutions (but dual feasible!), until a solution that is both primal and dual feasible is found. In our example, Table 3a is equivalent to the primal basic infeasible solution $\mathbf{x} = (0, 0, -3, -2)^T$. By complementary slackness, the associated dual solution is $\mathbf{u} = (0, 0)^T$, which is basic feasible in the dual problem. Figure 1, adapted from a similar example presented in [1], displays the progression of the dual simplex method in terms of primal and dual basic feasible solutions.

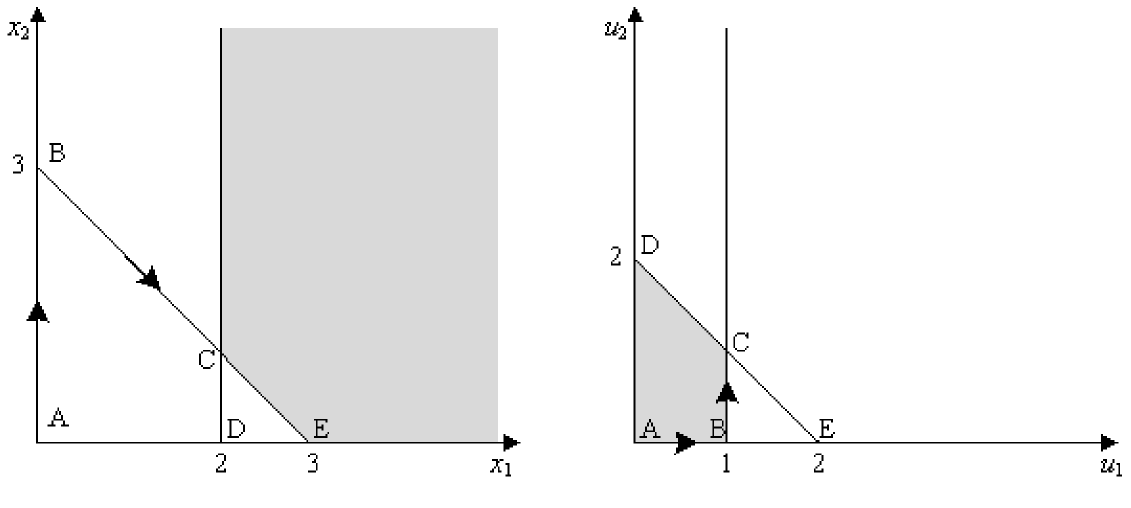

Figure 1: The primal and the dual feasible spaces

In Figure 1, the path A-B-C traces, in both spaces, the pivot sequence of the dual simplex applied to the example problem. In the primal space, bases A and B are infeasible, but at optimality, C is feasible. In the dual space, the algorithm maintains feasibility at each base A, B, and finally C. Naturally, the solution is improved during every iteration.

1.2 Implementation issues

1.2.1 Cycling

While iterating, it is possible that the dual simplex can arrive at a situation where the reduced cost associated with the selected pivot column is zero, a situation first identified in [12]. If this happens, then the reduced costs remain unchanged after pivoting, and therefore the value of the dual objective function ($\mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b}$) does not change as well. As a result, the dual simplex algorithm may cycle. The situation is avoided by choosing a lexicographic pivoting rule, which avoids cycling. Similar to anticycling rules for the primal simplex method, the rule states that one should choose the lexicographically smallest column, found by dividing all entries in the consideration set (columns with $a_{ik} < 0$) by a_{ik}. If a tie exists between several lexicographically smallest columns, one should choose the column with the smallest index. The proof of the finiteness of the simplex algorithm (either in primal or dual form) when using the lexicographic rule can be found in a number of reference texts on linear programming like [1], [13] or [14]. Besides the lexicographic rule, Bland's rule [15] (choose as entering variable the column with the smallest index that corresponds to a positive reduced cost and break ties among row candidates by favoring the row with the smallest index number) can also be easily adapted from the primal simplex algorithm and implemented in the dual simplex.

1.2.2 Degeneracy

Another issue that could appear while implementing the algorithm is dual degeneracy. Dual degeneracy appears whenever there are more than m reduced costs that are 0, that is, there exists at least a non-basic variable with reduced cost equal to zero. Since cycling and degeneracy are somewhat related, practical implementations of the dual simplex algorithm try to resolve both issues simultaneously, if possible (see, for example, the COIN-OR project that provides, among other things, open source implementations of simplex algorithms. A good overview of the project can be found in [16] and [17]).

1.2.3 Finding an initial dual basic feasible solution

As we have mentioned in the outline of the dual simplex algorithm, the algorithm must start with a basic dual feasible solution (BDFS), which sometimes is not readily available. Assuming that in the primal problem the initial basis is given by the first m columns, a dual feasible basis can be constructed by adding the constraint $\sum_{j=m+1}^{n} x_{ij} \leq M$ to the primal problem. Then, let the leaving variable be the slack measure associated with this constraint, and let the entering variable be the one corresponding to the maximum reduced cost. After the pivot, either the new basis is dual feasible and thus the dual simplex can commence, or the primal problem is optimal, or the primal is unbounded (see [11] for additional discussions as to why one of these outcomes must happen).

Example (Finding a BDFS: [11], pg. 277). Consider the following tableau, which we want to solve using dual simplex:

Table 4:

	z	x_1	x_2	x_3	x_4	x_5	RHS
z	1	0	1	5	-1	0	0
x_1	0	1	2	-1	1	0	4
x_5	0	0	3	4	-1	1	3

To get a dual basic feasible solution, we add the artificial constraint $x_2 + x_3 + x_4 \leq M$ to the problem. The slack of this constraint is x_6, and the resulting tableau is:

Table 5:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	1	5	-1	0	0	0
x_6	0	0	1	1^*	1	0	1	M
x_5	0	1	2	-1	1	0	0	4
x_1	0	0	3	4	-1	1	0	3

In the next step, x_6 leaves the basis, and x_3 enters (the maximum reduced cost is 5). The new tableau is dual feasible (but not primal feasible—notice the negative value of x_5).

Table 6:

	z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
z	1	0	-4	0	-6	0	-5	$-5M$
x_3	0	0	1	1	1	0	1	M
x_1	0	1	3	0	2	0	1	$M + 4$
x_5	0	0	-1	0	-5	1	-4	$-4M + 3$

While intuitively easy, this method is superseded in practical implementations by dual phase 1 algorithms. For an extensive review of dual phase 1 methods, see [18].

1.2.4 Selection Rules

From an implementation perspective, two important decisions that are quintessential to the practical performance of the dual simplex algorithm are the proper choices for the entering variable (also known as the “pricing” step) and the exiting variable (the “minimum ratio test” step). For a relatively long period of time, from the late 1950s until the early 1990s, the dual simplex was relatively ignored in practice, due to the perception that it would not outperform the primal simplex algorithm. The last important theoretical contribution from the 1950s is Wagner’s adaptation of the dual simplex for bounded variables [19]. However, as the understanding of the structure of linear programming problems grew, it has become increasingly obvious that there exist classes of programs for which the primal simplex method is better, and other types of problems that tend to be better solved by the dual simplex

[20]. [21] and [22] helped in changing the perception about the dual simplex algorithm, by devising clever pricing techniques (steepest edge) and better ratio tests (bound flipping ratio test), that boost the dual simplex performance over its primal counterpart. Separately, [23] generalized the two-phase primal simplex algorithm to its dual counterpart. An excellent survey of the progress made with respect to the practical implementation of the dual simplex algorithm can be found in [24]. Further breakthroughs emerged with the advance of parallelized implementations of the dual simplex [25].

2 Application of dual simplex: integer programming

One of the most important applications of the dual simplex is in large-scale optimization, particularly large-scale mixed integer programming. Traditionally, if the relaxation of a general mixed integer programming problem does not satisfy the original integrality requirements, then some mixture of variable branching and addition of valid inequalities happens, until the branch–and–bound/cut/price implementation converges to the optimal solution (assuming it exists). Notice that branching on a fractional variable creates two sub-problems, each of which has an additional constraint-a bound on the branching variable, which destroys the primal feasibility of the original problem. Similarly, the addition of a valid inequality to the problem breaks primal feasibility. However, in the dual space, the problem remains feasible, since the addition of a row in the primal is equivalent to the addition of a column to the dual. If the added column is non–basic, the dual basis can be used to continue the optimization via dual simplex, without the need to re–compute and re–factor a new basis (both of which are time intensive operations). This hot–starting feature makes the dual simplex very attractive for large-scale optimization projects. We illustrate this property with the following knapsack example:

Example (Valid Inequalities and Dual Simplex). Consider the following knapsack problem:

$$\begin{aligned} &\max 2x_1 + 3x_2 + 4x_3 \\ \text{subject to: } &x_1 + 2x_2 + 3x_3 \leq 4 \\ &x_1, x_2, x_3 \qquad \text{binary} \end{aligned}$$

The initial tableau for the relaxation is

Table 7:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
z	1	2	3	4	0	0	0	0	0
x_4	0	1	2	3	1	0	0	0	4
x_5	0	1	0	0	0	1	0	0	1
x_6	0	0	1	0	0	0	1	0	1
x_7	0	0	0	1	0	0	0	1	1

After several iterations we obtain the final optimal tableau for the LP relaxation

Table 8:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
z	1	0	0	0	$-\frac{4}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	$-\frac{19}{3}$
x_2	0	0	1	0	0	0	1	0	1
x_1	0	1	0	0	0	1	0	0	1
x_7	0	0	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{2}{3}$
x_3	0	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$	0	$\frac{1}{3}$

The solution is $x_1 = x_2 = 1, x_3 = \frac{1}{3}$. This solution does not satisfy the binary requirements. On the other hand, notice that x_2 and x_3 cannot be simultaneously equal to 1 in the optimal solution, as that would violate the knapsack constraint. Therefore, the inequality $x_2 + x_3 \leq 1$ is valid for the original knapsack problem, and can be added to the previous tableau, yielding

Table 9:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\boldsymbol{x_8}$	RHS
z	1	0	0	0	$-\frac{4}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	**0**	$-\frac{19}{3}$
x_2	0	0	1	0	0	0	1	0	**0**	1
x_1	0	1	0	0	0	1	0	0	**0**	1
x_7	0	0	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	1	**0**	$\frac{2}{3}$
x_3	0	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$	0	**0**	$\frac{1}{3}$
$\boldsymbol{x_8}$	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**

Subtracting row 1 and row 4 from row 5 in order to get $\mathbf{I}_5$ as basis, yields a dual feasible solution (notice that row 5 has a negative RHS, rendering the solution primal infeasible). Dual simplex leads to optimality after one pivot.

Table 10:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	RHS
z	1	0	0	0	$-\frac{4}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	0	$-\frac{19}{3}$
x_2	0	0	1	0	0	0	1	0	0	1
x_1	0	1	0	0	0	1	0	0	0	1
x_7	0	0	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	1	0	$\frac{2}{3}$
x_3	0	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$	0	0	$\frac{1}{3}$
x_8	0	0	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}^*$	0	1	$-\frac{1}{3}$
z	1	0	0	0	-1	$-\frac{2}{3}$	0	0	-1	-6
x_2	0	0	1	0	-1	1	0	0	3	0
x_1	0	1	0	0	0	1	0	0	0	1
x_7	0	0	0	0	-1	1	0	1	2	0
x_3	0	0	0	1	1	$-\frac{1}{3}$	0	0	-2	1
x_6	0	0	0	0	1	-1	1	0	-3	1

The solution is $x_1 = x_3 = 1, x_2 = 0$. Since this set also satisfies the binary requirements, it must be optimal to the original knapsack. The value of the objective function is $z = 6$.

3 Conclusions

Modern advances in linear programming theory establish the use of the dual simplex algorithm as a powerful optimization tool. While the performance of the dual simplex was originally considered to lag that of its most popular primal variant, the dual simplex is widely used today in practice. One of the more popular applications of the algorithm includes large-scale mixed integer programming, where row additions break primal feasibility, but typically, dual feasibility remains intact. The dual simplex is also used as a de-facto linear programming solver, since research has identified [20] many classes of problems where the application of the dual algorithm outperforms the performance of its primal counterpart. All these interesting properties of the method, coupled with the fact that linear optimization is a fundamental topic in operations research, ensure that advances in this area will continue to be positioned at the forefront of the discipline.

References

[1] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, MA, 1st edition, 1997.

[2] Katta G. Murty. *Linear Programming.* Wiley, New York, NY, 1st edition, 1983.

[3] George B. Dantzig. Programming in a linear structure, 1948.

[4] George B. Dantzig. Programming of interdependent activities ii: Mathematical model. *Econometrica*, 17(3/4):200–211, 1949.

[5] George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 359–373. John Wiley and Sons, Inc., New York, NY, 1951.

[6] C. E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1:36–47, 1954.

[7] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities - III*, pages 159–175. Academic Press, New York, NY, 1972.

[8] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[9] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[10] George B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1st edition, 1963.

[11] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows.* Wiley, New York, NY, 2nd edition, 1990.

[12] E. M. L. Beale. Cycling in the dual simplex algorithm. *Naval Research Logistics Quarterly*, 2(4):269–275, 1955.

[13] Vašek Chvátal. *Linear Programming.* Freeman, 1st edition, 1983.

[14] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer, 2nd edition, 2001.

[15] Robert G. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977. doi: 10.1287/moor.2.2.103.

[16] Robin Lougée-Heimer, Francisco Barahona, Brenda Dietrich, J. P. Fasano, John Forrest, Robert Harder, Laszlo Ladanyi, Tobias Pfender, Theodore Ralphs, Matthew Saltzman, and Katya Scheinberg. The coin-or initiative: Open-source software accelerates operations research progress. *ORMS Today*, 28(5):20–22, 2001.

[17] Robin Lougée-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[18] Achim Koberstein and Uwe Suhl. Progress in the dual simplex method for large scale lp problems: practical dual phase 1 algorithms. *Computational Optimization and Applications*, 37(1):49–65, 2007.

[19] H. M. Wagner. The dual simplex algorithm for bounded variables. *Naval Research Logistics Quarterly*, 5:257–261, 1958.

[20] Robert E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(2):3–15, 2002.

[21] J. J. Forrest and D. Goldfarb. Steepest edge simplex algorithms for linear programming. *Mathematical Programming*, 57(3):341–374, 1992.

[22] R. Fourer. Notes on the dual simplex method. Technical report, Draft Report, Northwestern University, 1994.

[23] I. Maros. A generalized dual phase-2 simplex algorithm. *European Journal of Operational Research*, 149(1):1–16, 2003.

[24] Achim Koberstein. Progress in the dual simplex algorithm for solving large scale lp problems: techniques for a fast and stable implementation. *Computational Optimizations and Applications*, 41(2):185–204, 2008.

[25] R. E. Bixby and A. Martin. Parallelizing the dual simplex method. *INFORMS Journal on Computing*, 12(1):45–56, 2000.