A Generalized Dual Phase-2 Simplex Algorithm[1]

István Maros

Department of Computing, Imperial College, London

Email: i.maros@ic.ac.uk

Departmental Technical Report 2001/2

ISSN 1469–4174

January 2001, revised December 2002

[1]This research was supported in part by EPSRC grant GR/M41124.

Contents

Abstract

Recently, the dual simplex method has attracted considerable interest. This is mostly due to its important role in solving mixed integer linear programming problems where dual feasible bases are readily available. Even more than that, it is a realistic alternative to the primal simplex method in many cases. Real world linear programming problems include all types of variables and constraints. This necessitates a version of the dual simplex algorithm that can handle all types of variables efficiently. The paper presents increasingly more capable dual algorithms that evolve into one which is based on the piecewise linear nature of the dual objective function. The distinguishing features of this method are: (i) in one iteration it can make progress equivalent to many traditional dual iterations, (ii) using proper data structures it can be implemented very efficiently so that an iteration requires hardly more work than the traditional pivot method, (iii) its effectiveness just increases if more upper bounded variables are present, (iv) it has inherently better numerical stability because it can create a large flexibility in finding a pivot element, (v) it can excel itself in coping with degeneracy as it can bypass dual degenerate vertices more easily than the traditional pivot procedures. The power of the method is demonstrated through examples. The performance on some real world problems is also presented.

1 Introduction

Not long after the publication of Dantzig's primal simplex algorithm [3] (in 1951) its dual version, developed by Lemke [7], also appeared (in 1954). It has long been known that the dual simplex algorithm (DSA) is a better alternative to the primal simplex for solving certain types of linear programming (LP) problems. Its importance has been recognized in many areas, in particular in the case of the Branch and Bound (BB) type methods for Mixed Integer Linear Programming (MILP). BB requires the repeated solution of closely related continuous LP problems (c.f. Nemhauser and Wolsey [11]). At each branching node two new subproblems (child problems) are created that differ from the parent problem in the individual bound of the branching variable. Usually the right-hand-sides of the child problems also change. The result is that the basis of the parent problem becomes primal infeasible for the derived problems but it remains (or can easily be made) dual feasible. This makes the dual simplex method an obvious choice for the required 'reoptimization'.

In real world LP problems all sorts of variables and constraints can be present. The standard dual algorithm has been worked out for the case where all variables are nonnegative and a dual feasible basis is available.

The LP relaxation of a MILP problem can contain many bounded variables. Free and fixed variables may also be present. The increasing need for solving various types of LP problems with the dual simplex method necessitates the design and implementation of a version of the DSA where variables of arbitrary type are allowed and they all, especially the bounded primal variables, are treated efficiently. The main purpose of this paper is to present such an algorithm. This is achieved by discussing increasingly more capable dual algorithms that evolve into one that has the potential of this efficiency.

The rest of the paper is organized as follows. In section 2 the primal and dual problems are stated and an overview of the work done in this area is given. Issues of dual feasibility are discussed in section 3. Section 4 presents increasingly more capable dual algorithms including one that can handle all types of variables. The background of a new algorithm is introduced in section 5 while section 6 gives its step-by-step description followed by a

detailed analysis of the main features. In section 7 two examples are given that demonstrate the power of the algorithm. While this paper is not meant to be a computational study, some limited computational experience with real world problems is presented in section 8. It is followed by a brief summary in section 9.

2 Problem statement

First, we introduce some notations. Matrices will be denoted by boldface capitals, like $\mathbf{A}$, vectors by boldface lower case Latin or Greek letters, like $\mathbf{b}$ or $\boldsymbol{\beta}$. Column j of matrix $\mathbf{A}$ is $\mathbf{a}_j$ while row i of $\mathbf{A}$ is $\mathbf{a}^i$. Elements of matrices and vectors are denoted by the respective normal lower case letters. Matrix element at the intersection of row i and column j is denoted by a^i_j.

2.1 The primal problem

We consider the following primal linear programming problem:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x}, \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{aligned} \tag{1}$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$, $\mathbf{x}$, $\mathbf{l}$, and $\mathbf{u}$ are n-vectors, and $\mathbf{b}$ is an m vector. Any component of $\mathbf{l}$ and $\mathbf{u}$ can be $-\infty$ or $+\infty$, respectively. $\mathbf{A}$ is assumed to contain a unit matrix $\mathbf{I}$, that is, $\mathbf{A} = [\mathbf{I}, \hat{\mathbf{A}}]$. Variables that multiply columns of $\mathbf{I}$ transform every constraint to an equation and are often referred to as *logical variables*. Variables which multiply columns of $\hat{\mathbf{A}}$ are called *structural variables*.

After translation it can be achieved that variables (whether logical or structural) fall into four categories as shown below (for further details, see Orchard-Hays [12]).

Feasibility range	Type	Reference
$x_j = 0$	0	Fixed variable
$0 \leq x_j \leq u_j < +\infty$	1	Bounded variable
$0 \leq x_j \leq +\infty$	2	Non-negative variable
$-\infty \leq x_j \leq +\infty$	3	Free variable

(2)

2.2 The dual problem

First, we investigate the primal problem where all variables are bounded.

$$\begin{array}{lll} (P1) & \text{minimize} & \mathbf{c}^T\mathbf{x}, \\ & \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{array} \tag{3}$$

Since $\mathbf{A}$ is of full row rank m independent columns can be selected to form a basis to (P1). Let B denote the index set of basic and N the index set of nonbasic variables. Accordingly, we use notation $\mathbf{c}_B$, $\mathbf{c}_N$ and $\mathbf{x}_B$, $\mathbf{x}_N$ to refer to the basic and nonbasic components of $\mathbf{c}$ and $\mathbf{x}$, respectively. The basis matrix itself is denoted by $\mathbf{B}$ and the rest of $\mathbf{A}$ by $\mathbf{N}$. A basic solution is

$$\mathbf{x}_B = \mathbf{B}^{-1}\left(\mathbf{b} - \sum_{j \in U} u_j \mathbf{a}_j\right),$$

where U is the index set of nonbasic variables at upper bound. The ith basic variable is denoted by x_{Bi}. The d_j reduced cost of variable x_j is defined as

$$d_j = c_j - \boldsymbol{\pi}^T\mathbf{a}_j = c_j - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_j \tag{4}$$

which is further equal to $c_j - \mathbf{c}_B^T\boldsymbol{\alpha}_j$ if notation $\boldsymbol{\alpha}_j = \mathbf{B}^{-1}\mathbf{a}_j$ is used. The dual of (P1) is:

$$\begin{array}{lll} (D1) & \text{maximize} & \mathbf{b}^T\mathbf{y} - \mathbf{u}^T\mathbf{w}, \\ & \text{subject to} & \mathbf{A}^T\mathbf{y} - \mathbf{w} \leq \mathbf{c}, \\ & & \mathbf{w} \geq \mathbf{0} \end{array} \tag{5}$$

In case of *Zero-One Mixed Integer Programming*, some or all of the variables may be restricted to be 0 or 1 (*0/1 variables*). It means that in the LP relaxation there are many variables with upper bound of 1.

Branch and Bound type algorithms require the solution of many closely related LP problems. Since the optimal basis of the parent problem usually remains dual feasible for the child problems, the dual algorithm can be used efficiently for 'reoptimization' starting from this basis.

In practice, dual algorithms work on the primal problem using the computational tools of the sparse primal simplex method but perform basis changes according to the rules of the dual.

An upper bounded version of the DSA was first described by Wagner [13] and later by Orchard-Hays [12] and Chvátal [2]. One of the computationally most expensive steps of the algorithm is the creation of the updated pivot row p, i.e., the computation of α_j^p for all nonbasic j (c.f. [10]). None of the quoted descriptions discusses the multiple use of the updated pivot row which becomes possible in case of the presence of upper bounded variables. In the BSD (for Bound Swapping Dual) algorithm to be presented in section 5 we concentrate on this issue and show how it may be possible to make—occasionally many—iterations with one updated pivot row.

The underlying idea comes from the FEWPHI algorithm [8] of the author that can make several primal iterations with one updated column. Similar ideas for the primal were also discussed by Wolfe [14] and Greenberg [6]. Fourer in [5] described the dual in a similar vein. A reference by Bixby et al. to the usefulness of 'long step' dual methods can be found in [1]. The present paper elaborates on the ideas of bound swapping presented by Maros in [9] and gives a detailed discussion of the dual with all types of variables.

3 Dual feasibility

If there are only type-2 variables in the problem then $\mathbf{w}$ is not present in (5). If $\mathbf{B}$ is a basis then dual feasibility reduces to

$$\begin{aligned} \mathbf{B}^T\mathbf{y} &= \mathbf{c}_B \\ \mathbf{N}^T\mathbf{y} &\leq \mathbf{c}_N, \quad \text{or} \quad \mathbf{d}_N = \mathbf{c}_N - \mathbf{N}^T\mathbf{y} \geq 0, \end{aligned}$$

which is the primal optimality condition for the problem.

The idea of a dual step is based on the relaxation of the p-th dual basic equation from $\mathbf{a}_p^T\mathbf{y} = c_p$ to $\mathbf{a}_p^T\mathbf{y} \leq c_p$. We say that this constraint is relaxed negatively because $\mathbf{a}_p^T\mathbf{y}$ is decreased. If the change in the p-th equation is parameterized by $t \leq 0$ then we have $\mathbf{B}^T\mathbf{y}(t) - t\mathbf{e}_p = \mathbf{B}^T\mathbf{y}$, from which the dual solution as a function of t is $\mathbf{y}(t) = \mathbf{y} + t\boldsymbol{\rho}_p$, if $\boldsymbol{\rho}_p$ denotes the p-th column of the transpose of the inverse $\mathbf{B}^{-T}$. The corresponding dual objective value is:

$$f(t) = \mathbf{b}^T\mathbf{y}(t) = \mathbf{b}^T\mathbf{y} + t\mathbf{b}^T\boldsymbol{\rho}_p = \mathbf{b}^T\mathbf{y} + tx_{Bp} \tag{6}$$

As t moves away from 0 negatively the rate of change in the dual objective is $-x_{Bp}$ which leads to an improvement over $\mathbf{b}^T\mathbf{y}$ if $x_{Bp} < 0$. This suggests that $x_{Bp} < 0$ (an infeasible primal basic variable) has to be selected to leave the basis at a feasible level (usually 0). The entering variable is determined by the dual ratio test to ensure that dual feasibility ($\bar{\mathbf{d}}_N \geq \mathbf{0}$) is maintained after the basis change. The dual constraint corresponding to the entering variable replaces the relaxed constraint as the new tight one.

If there are all types of variables present in a problem it remains true that dual feasibility is equivalent to primal optimality. In this case dual feasibility means that for

nonbasic positions d_j of (4) satisfies

type(x_j)	Status	d_j
0	$x_j = 0$	Immaterial
1	$x_j = 0$	≥ 0
1	$x_j = u_j$	≤ 0
2	$x_j = 0$	≥ 0
3	$x_j = 0$	$= 0$

(7)

Let r denote the original index of the p-th basic variable ($x_{Bp} \equiv x_r$). According to the transformation formulae of the simplex method, if $x_q,\ q \in N$, is the incoming and x_{Bp} is the outgoing variable, the d_js are transformed as

$$\begin{aligned} \bar{d}_j &= d_j - \frac{d_q}{\alpha_q^p}\,\alpha_j^p,\ j \in N \\ \bar{d}_r &= -\frac{d_q}{\alpha_q^p}. \end{aligned} \tag{8}$$

The purpose of the dual step, whence x_{Bp} has been selected, is to choose q in such a way that dual feasibility is preserved. This can be achieved by an appropriately designed dual ratio test. From (7) it is clear that the d_j of type-0 variables is not restricted in sign, therefore such positions do not take part in the ratio test. It is also a requirement that x_{Bp} leave the basis in such a way that $\bar{d}_r$ also satisfies (7). Note, the feasibility of the primal basic variables is not explicitly taken into account. Therefore, after a basis change their number and magnitude are unknown. Even the incoming variable can enter the basis at infeasible level. These ideas form the basis of different dual algorithms.

4 Dual simplex methods

In this section we give the algorithmic description of several dual simplex methods that have been developed for different types of problems. They are increasingly more capable and will form the basis of the new method.

4.1 Traditional algorithm

Lemke's original version of the simplex method was developed for the standard LP problem when only type-2 variables are present, i.e., $\min \mathbf{c}^T\mathbf{x},\ \mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$.

Assume a dual feasible basis $\mathbf{B}$ is available i.e., $\mathbf{d}_N \geq \mathbf{0}$, together with $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$. For this case the dual algorithm, which we call *Dual-1*, can be stated in the framework of the revised simplex method as follows:

Step 1. If the current basic solution is primal feasible, $\mathbf{x}_B \geq \mathbf{0}$, then it is optimal. Otherwise, select an infeasible basic variable, say $x_{Bp} < 0$. This variable leaves the basis, row p is the pivot row.

Step 2. Determine the nonbasic components of the updated pivot row: $\boldsymbol{\alpha}_N^p = \mathbf{e}_p^T\mathbf{B}^{-1}\mathbf{N}$. In coordinate form it is $\alpha_j^p = \boldsymbol{\rho}_p^T\mathbf{a}_j,\ j \in N$, where $\boldsymbol{\rho}_p^T$ is the p-th row of $\mathbf{B}^{-1}$, i.e., $\boldsymbol{\rho}_p^T = \mathbf{e}_p^T\mathbf{B}^{-1}$ with $\mathbf{e}_p$ being the p-th unit vector in $\mathbb{R}^m$.

Step 3. Based on (7) and (8), determine the index set of eligible pivot positions $J = \{j : \alpha_j^p < 0,\ j \in N\}$. If J is empty then stop: the dual problem is unbounded, consequently, the primal has no feasible solution.

Otherwise, perform dual ratio test with $\mathbf{d}_N$ and $\boldsymbol{\alpha}_N^p$ to determine subscript q of the incoming variable:

$$\theta^{(d)} = -\frac{d_q}{\alpha_q^p} = \min_{j \in J}\left\{-\frac{d_j}{\alpha_j^p}\right\}.$$

$\theta^{(d)}$ is the minimum dual ratio.

Step 4. Determine the transformed entering column: $\boldsymbol{\alpha}_q = \mathbf{B}^{-1}\mathbf{a}_q$.

Step 5. Update solution.

Let r be the original index of x_{Bp} in $\mathbf{A}$, i.e., $x_{Bp} \equiv x_r$. Set $\bar{x}_r = 0$ and $\theta^{(p)} = \dfrac{x_{Bp}}{\alpha_q^p}$ (the corresponding primal 'ratio').

Update the basic solution $\bar{x}_{Bi} = x_{Bi} - \theta^{(p)}\alpha_q^i$ for $1 \leq i \leq m,\ i \neq p$ and $\bar{x}_{Bp} = \theta^{(p)}$. In this case the incoming variable enters at feasible level.

Update reduced cost of nonbasic variables: $\bar{d}_r = \theta^{(d)}$ and $\bar{d}_j = d_j + \theta^{(d)}\alpha_j^p,\ j \neq r$.

Go to Step 1.

4.2 Dual algorithm with type-1 variables present

If there are type-1 variables in the LP problem then dual feasibility for such variables means that $d_j \geq 0$ if x_j is nonbasic at zero and $d_j \leq 0$ if x_j is nonbasic at upper bound. Additionally, an upper bounded basic variable can be infeasible in two ways, either $x_{Bi} < 0$ or $x_{Bi} > u_{Bi}$.

Following Chvátal [2, p. 157], the steps of an upper bounded dual algorithm that solves (3), which we call *Dual-2*, can be described in the following way (for further assumptions and notations, see Dual-1).

Step 1. If the corresponding primal solution is feasible, that is, $\mathbf{0} \leq \mathbf{x}_B \leq \mathbf{u}_B$ then it is optimal. Otherwise, there is at least one primal infeasible basic variable (see remark above). Any such variable can be selected to leave the basis. Let the basic position of the chosen one be denoted by p.

Step 2. Determine the nonbasic components of the updated pivot row: $\boldsymbol{\alpha}_N^p = \boldsymbol{\rho}_p^T\mathbf{N}$. Identical with Step 2 of Dual-1.

Step 3. The index set of eligible pivot positions is determined by (7) and (8). If $x_{Bp} < 0$ then let $J_1 = \{j : \alpha_j^p < 0,\ x_j = 0,\ j \in N\}$ and $J_2 = \{j : \alpha_j^p > 0,\ x_j = u_j,\ j \in N\}$. If $x_{Bp} > u_{Bp}$ then let $J_1 = \{j : \alpha_j^p > 0,\ x_j = 0,\ j \in N\}$ and $J_2 = \{j : \alpha_j^p < 0,\ x_j = u_j,\ j \in N\}$. In both cases, let $J = J_1 \cup J_2$.

If J is empty then stop: the problem is dual unbounded and, hence, primal infeasible. Otherwise, find $q \in J$ such that

$$\theta^{(d)} = \left|\frac{d_q}{\alpha_q^p}\right| = \min_{j \in J}\left|\frac{d_j}{\alpha_j^p}\right| \tag{9}$$

and let x_q be the entering variable.

Step 4. Determine the transformed entering column: $\boldsymbol{\alpha}_q = \mathbf{B}^{-1}\mathbf{a}_q$. Identical with Step 4 of Dual-1.

Step 5. Update solution.

Set $\theta^{(p)} = \dfrac{x_{Bp}}{\alpha_q^p}$ and $\bar{x}_r = 0$ in case $x_{Bp} < 0$ or $\theta^{(p)} = \dfrac{x_{Bp} - u_{Bp}}{\alpha_q^p}$ and $\bar{x}_r = u_r$ in case $x_{Bp} > u_{Bp}$.

Update the basic solution: $\bar{x}_{Bi} = x_{Bi} - \theta^{(p)}\alpha_q^i$ for $1 \leq i \leq m,\ i \neq p$ and

$$\bar{x}_{Bp} = x_q + \theta^{(p)}.$$

Now the incoming variable may enter at infeasible level. For details, see section 5.

Update reduced cost of nonbasic variables: $\bar{d}_r = -\dfrac{d_q}{\alpha_q^p}$ and $\bar{d}_j = d_j + \bar{d}_r\alpha_j^p,\ j \neq r$.

Go to Step 1.

4.3 Dual algorithm with all types of variables

Nonbasic type-0 variables do not play any role in the dual method since any d_j is feasible for a fixed variable x_j. Such variables do not take part in the dual ratio test (are not included in J) and, therefore, they never enter the basis. This is completely in line with the selection principles of the primal simplex algorithm. If a type-0 variable is basic (e.g., the logical variable of an equality constraint) it can be feasible (i.e., equal to zero) or infeasible (positive of negative). Such an infeasible variable is always a candidate to leave the basis and thus improve the dual objective as it can be seen from (6). Since the reduced cost of the outgoing type-0 variable is unrestricted in sign it can always be chosen to point to the improving direction of the dual objective function.

The way type-1 variables can be handled has already been discussed. In many, but not all, aspects type-0 variables can be viewed as special cases of type-1 when $l_j = u_j = 0$. The main difference is that once a type-0 variable left the basis it will never be a candidate to enter again.

Lemke's original DSM was developed for the case of type-2 variables. Such variables can be treated in a way described in Dual-1.

Type-3 (free) variables can also be present in an LP problem. If they are nonbasic dual feasibility requires $d_j = 0$ for them. If they are basic they are always at feasible level and are not candidates to leave the basis.

It is easy to see that *Dual-2* algorithm requires very few changes to take care of all types of variables. Type-0 variables are of interest only if they are basic at infeasible level. At the other extreme, type-3 variables are involved in the dual algorithm as long as they are nonbasic. In this case they need to be included in the ratio test as their d_j must remain zero. They will certainly take part in the (9) dual ratio test if the corresponding $\alpha_j^p \neq 0$. Since in this case the ratio is zero such a variable will be selected to enter the basis. This, again, is very similar to the tendency of the primal method which also favors free variables to become basic if their reduced cost is nonzero.

For the formal statement of the algorithm with all types of variables, which we refer to as *Dual-3*, only Step 3 of Dual-2 has to be adjusted. It is important to remember that now the basis may contain some type-0 variables for which both lower and upper bound are zero. Therefore, such basic variables can be infeasible positively or negatively the same way as type-1 variables. If selected, a type-0 variable will leave the basis at zero level but the sign of its d_j will become unimportant after the iteration as it is unrestricted.

Step 3. Let J_1 be the set of indices of type-3 nonbasic variables for which $\alpha_j^p \neq 0$.

For type-1 and type-2 nonbasic variables:

If $x_{Bp} < 0$ then let J_2 be the index set of those x_js for which $\alpha_j^p < 0$ and $x_j = 0$, or $\alpha_j^p > 0$ and $x_j = u_j$. If $x_{Bp} > u_{Bp}$ then let J_2 be the set of those x_js for which $\alpha_j^p > 0$ and $x_j = 0$, or $\alpha_j^p < 0$ and $x_j = u_j$.

Let $J = J_1 \cup J_2$.

If J is empty then stop. The problem is dual unbounded and, hence, primal infeasible.

Otherwise, find $q \in J$ such that

$$\theta^{(d)} = \left|\frac{d_q}{\alpha_q^p}\right| = \min_{j \in J} \left|\frac{d_j}{\alpha_j^p}\right| \tag{10}$$

and let x_q be the entering variable ($\theta^{(d)}$ is the selected dual ratio).

5 Bound swap in dual

While Dual-2 and Dual-3 are perfectly legitimate algorithms for the case when upper bounded variables are present in a problem further investigations reveal some additional possibilities that can be computationally very appealing.

At the beginning of an iteration the incoming variable x_q is nonbasic at a feasible level. It becomes basic in position p with a value of $x_q + \theta^{(p)}$ which will be infeasible if it is a bounded variable and its displacement is greater than its own individual bound, i.e.,

$$|\theta^{(p)}| > u_q. \tag{11}$$

Though this situation does not change the theoretical convergence properties of the dual algorithm (no basis is repeated if dual degeneracy is treated properly), computationally it is usually not advantageous.

The introduction of a new infeasible basic variable can be avoided. One possibility is to set x_q to its opposite bound (bound swap). It entails the updating of the basic solution. However, the corresponding dual variable becomes infeasible now (see (7)). At the same time, the updated x_{Bp} remains infeasible and can be selected again. The ratio test can be performed at no cost as the updated pivot row remains available. This step can be repeated recursively until a variable is found that enters the basis at a feasible level or we can conclude that the dual solution is unbounded. In the former case the dual feasibility of variables participating in bound change is automatically restored. A careful analysis shows that this idea leads to an algorithm that has some remarkable features. One of them is that its efficiency just increases with the number of upper bounded variables. A more complete summary of features is given in section 6.5.

As a bounded (type-0 or type-1) basic variable can be infeasible in two different ways, first we discuss these cases separately then present a unifying framework.

First, we assume that p was selected because $x_{Bp} < 0$. In this case the p-th dual

constraint is released negatively and the change of the objective function is described by (6). Now the ratios in (10) will be negative. If, after applying this ratio test, (11) holds for the resulting $\theta^{(p)}$ then a bound swap of x_q is triggered and the $\mathbf{x}_B$ basic solution is updated as $\bar{\mathbf{x}}_B = \mathbf{x}_B \pm u_q \boldsymbol{\alpha}_q$. More specifically, the selected infeasible basic variable gets updated as $\bar{x}_{Bp} = x_{Bp} \pm u_q \alpha_q^p$. Here, '$-$' applies if $x_q = 0$, and '$+$' if $x_q = u_q$ before the iteration. Taking into account the definition of J (see Step 3 of Dual-3) the two cases can be expressed by the single equation:

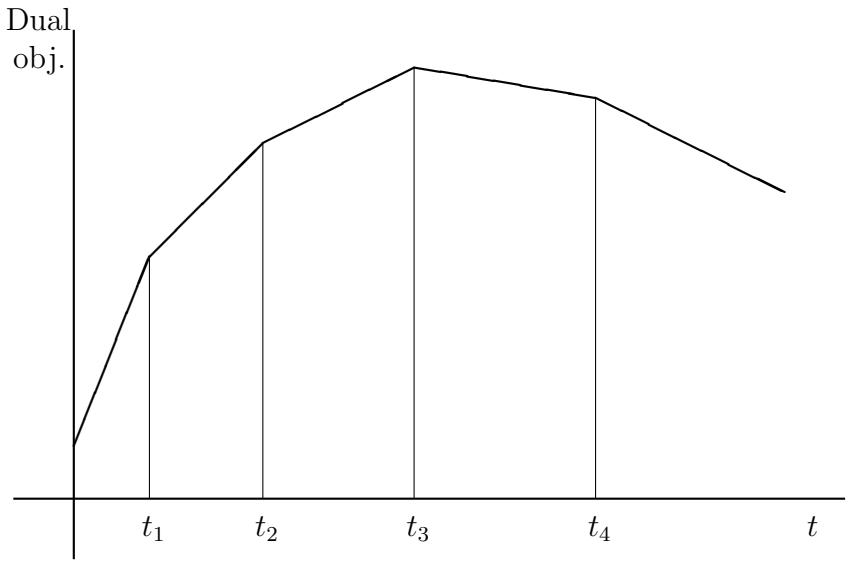
$$\bar{x}_{Bp} = x_{Bp} + |\alpha_q^p| u_q.$$

Here, $\bar{x}_{Bp} < 0$ holds because of $|x_{Bp}/\alpha_q^p| > u_q$, meaning that the originally selected infeasible basic variable remains infeasible. It implies that the dual objective function keeps improving if t moves beyond $\theta^{(d)}$, however, the rate of improvement decreases by $|\alpha_q^p| u_q$ from $-x_{Bp}$ to $-x_{Bp} - |\alpha_q^p| u_q$. Now, we leave out index q from J, go to the (10) ratio test, and repeat the above procedure. This can easily be done since the updated p-th row remains available unchanged. If with the newly defined $\theta^{(p)}$ we find $|\theta^{(p)}| \leq u_q$ then the corresponding x_q is the incoming variable and a basis change is performed. If $\bar{x}_{Bp} < 0$ holds and J becomes empty then the dual is unbounded and the primal is infeasible.

Figure 1 shows the change of the dual objective as a function of relaxing the p-th dual constraint. This is a piecewise linear function of t. The first breakpoint corresponds to the smallest ratio the second to the second smallest, and so on. At the maximum the sign of the slope of the function changes. It is determined by $|\theta^{(p)}| \leq u_q$, that is, when a basis change is performed.

Next, we consider the case when p was selected because $x_{Bp} > u_{Bp}$. Any constraint $0 \leq x \leq u$ defining an upper bounded variable can be written as $x + v = u,\ x, v \geq 0$, where v is a slack variable. Therefore, the $x_{Bp} > u_{Bp}$ condition is equivalent to $v_{Bi} = u_{Bi} - x_{Bi} < 0$. This latter indicates that if the signs of the entries in row p are changed we have the same situation as before. Therefore, by relaxing the p-th dual constraint positively to $\mathbf{a}_p^T \mathbf{y} \geq c_p$ (which is now parameterized by $t \geq 0$) the change in the dual objective is $\Delta_{d\_obj} = -t v_{Bi} = t(x_{Bp} - u_{Bp})$, that is, the rate of change is $x_{Bp} - u_{Bp} > 0$ as t moves

Figure 1: The dual objective as the function of releasing a dual basic equation.



away from 0 positively. The $\mathbf{a}_p^T\mathbf{y} \geq c_p$ type relaxation means that the corresponding primal variable must leave the basis at upper bound (see (7)).

Similarly to the first case, (10) is performed with the appropriate J. Note, the ratios are positive now. A bound swap of x_q is triggered if (11) holds and the updated value of the p-th basic variable becomes $\bar{x}_{Bp} = x_{Bp} \pm u_q\alpha_{pq}$. Here, '+' applies if $x_q = 0$, and '$-$' if $x_q = u_q$ before the iteration, but in this case (see definition of J in Dual-3) the single updating equation is

$$\bar{x}_{Bp} = x_{Bp} - |\alpha_q^p|u_q.$$

Now, $\bar{x}_{Bp} > u_{Bp}$ holds because of $\theta^{(p)} = |(x_{Bp} - u_{Bp})/\alpha_q^p| > u_q$, meaning that the originally selected infeasible basic variable remains infeasible. It implies that the dual objective function keeps improving if t is increased beyond $\theta^{(d)}$, however, the rate of improvement decreases again by $|\alpha_q^p|u_q$. The rest of the arguments of the first case applies here unchanged.

The important common feature of the two cases is that the slope of the corresponding piecewise linear function (the dual objective) decreases at every bound swap by $|\alpha_q^p|u_q$.

In other words, the initial slope of the dual objective is

$$s_0 = \begin{cases} -x_{Bp}, & \text{if } x_{Bp} < 0 \\ x_{Bp} - u_{Bp}, & \text{if } x_{Bp} > u_{Bp}. \end{cases} \tag{12}$$

This slope changes at the k-th bound swap to

$$s_k = s_{k-1} - |\alpha_q^p| u_q \tag{13}$$

(using the most recently defined q) until a basis change is encountered.

There can be several identical ratios. They represent the same breakpoint (which is called a *multiple breakpoint*) on the graph but they contribute to the decrease of the slope as if they were distinct. The multiplicity of a breakpoint is the number of identical ratios defining it. If the maximum of $f(t)$ is attained at a multiple breakpoint then dual degeneracy is generated after the iteration.

The change of the dual objective function can easily be traced. First, the absolute values of the ratios need to be sorted in ascending order, $0 \le |t_1| \le \cdots \le |t_Q|$, where $Q = |J|$. Let f_k denote the value of the dual objective at breakpoint k and $f_0 = \mathbf{b}^T\mathbf{y}$ (the value at the beginning of the iteration). Defining $t_0 = 0$ and following (6), f_k can be computed as:

$$f_k = f_{k-1} + (t_k - t_{k-1}) s_{k-1}, \quad k = 1, \ldots, Q.$$

6 Bound Swapping Dual (BSD) algorithm

Based on the observations of the previous section now we can define the following Bound Swapping Dual (BSD) algorithm for solving a general LP problem (as defined in (1)) with all types of variables present (as in (2)).

6.1 Step-by-step description of BSD

Assumptions of BSD: a dual feasible basis $\mathbf{B}$ to the problem with $\mathbf{d}_N$ satisfying (7) and the primal solution $\mathbf{x}_B = \mathbf{B}^{-1}\left(\mathbf{b} - \sum_{j \in U} u_j \mathbf{a}_j\right)$ are available.

Step 1. If $\mathbf{x}_B$ is primal feasible, i.e., satisfies (2) then it is optimal. Otherwise, select an infeasible basic variable and denote its basic position by p (pivot row).

Step 2. Determine the nonbasic components of the updated pivot row: $\boldsymbol{\alpha}_N^p = \mathbf{e}_p^T \mathbf{B}^{-1}\mathbf{N}$ as in Step 2 of Dual-1.

Step 3. Determine the breakpoints of $f(t)$ by computing dual ratios for eligible positions (set J as defined in Step 3 of Dual-3). Store their absolute values in a sorted order: $0 \leq |t_1| \leq \cdots \leq |t_Q|$.

If $Q = 0$ the problem is dual unbounded (primal infeasible), stop.

Otherwise set $k = 1$, $T^+ = T^- = \emptyset$ and s_0 according to (12).

Step 4. Let q denote the subscript of the variable defining t_k. If for the corresponding $\theta^{(p)}$ (as defined in Step 5 of Dual-2) $|\theta^{(p)}| \leq u_q$ then entering variable x_q is found, go to Step 5. (Clearly, if type of x_q is 2 or 3 then x_q is the incoming variable as the slope in (13) becomes $-\infty$ because of $u_q = \infty$.)

Otherwise, a bound swap is triggered. In accordance with its direction, set

$x_q = u_q$ or $x_q = 0$,

$T^+ := T^+ \cup \{q\}$ or $T^- := T^- \cup \{q\}$.

Compute $f_k = f_{k-1} + (t_k - t_{k-1})s_{k-1}$ and $s_k = s_{k-1} - |\alpha_q^p| u_q$. If $s_k > 0$ and $k < Q$ increment k and go to the beginning of Step 4.

If $k = Q$ and s_k is still positive then dual is unbounded, stop.

Step 5. Update solution:

1. Take care of bound swaps:

$$\bar{\mathbf{x}}_B = \mathbf{x}_B - \sum_{j \in T^+} u_j \boldsymbol{\alpha}_j + \sum_{j \in T^-} u_j \boldsymbol{\alpha}_j, \tag{14}$$

where $\boldsymbol{\alpha}_j = \mathbf{B}^{-1}\mathbf{a}_j$ and the sum is defined to be 0 if the corresponding index set is empty.

2. Take care of basis change:

   Replace $\bar{\mathbf{x}}_B$ by $\mathbf{F}\bar{\mathbf{x}}_B$, $\mathbf{F}$ denoting the elementary transformation matrix created from $\boldsymbol{\alpha}_q$. Let $x_{Bp} = x_q$. Update x_{Bp} to become $x_q + \theta^{(p)}$ and transform the reduced costs as defined in Step 5 of Dual-2.

Verbally, BSD can be interpreted in the following way: If not the smallest ratio is selected for pivoting then the d_js of the bypassed small ratios change sign. It can be compensated by putting the bypassed variables to their opposite bounds and thus maintaining the dual feasibility of the solution.

6.2 Work per iteration

It can easily be seen that the extra work required for BSD is generally small.

1. Ratio test: the same as in Dual-3.

2. The breakpoints of the piecewise linear dual objective function have to be stored and sorted. This requires extra memory for the storage, and extra work for the sorting. However, the t_k values have to be sorted only up to the point when a basis change is defined. Therefore, if an appropriate priority queue is set up for these values the extra work can be reduced sharply.

3. Taking care of bound swap according to (14) would require multiple update (FTRAN) involving all participating columns. However, the same can be achieved by only one extra FTRAN because

$$\begin{aligned}\bar{\mathbf{x}}_B &= \mathbf{x}_B - \sum_{j\in T^+} u_j\boldsymbol{\alpha}_j + \sum_{j\in T^-} u_j\boldsymbol{\alpha}_j \\ &= \mathbf{x}_B - \mathbf{B}^{-1}\left(\sum_{j\in T^+} u_j\mathbf{a}_j - \sum_{j\in T^-} u_j\mathbf{a}_j\right) \\ &= \mathbf{x}_B - \mathbf{B}^{-1}\tilde{\mathbf{a}}\end{aligned}$$

   with the obvious interpretation of $\tilde{\mathbf{a}}$.

6.3 Degeneracy

In case of dual degeneracy, we have $d_j = 0$ for one or more nonbasic variables. If these positions take part in the ratio test they define 0 ratios. This leads to a (multiple) break point at $t = 0$. Choosing one of them results in a non-improving iteration. Assume the multiplicity of 0 is ℓ, i.e.,

$$0 = |t_1| = \cdots = |t_\ell| < |t_{\ell+1}| \leq \cdots \leq |t_Q| \tag{15}$$

Denote the corresponding coefficients in the updated pivot row by $\alpha_{j_1}, \ldots, \alpha_{j_\ell}, \ldots, \alpha_{j_Q}$ (omitting the superscript p, for simplicity). The maximizing break point is defined by subscript k such that $s_k > 0$ and $s_{k+1} \leq 0$, i.e.,

$$s_k = s_0 - \sum_{i=1}^{k} |\alpha_{j_i}| > 0 \quad \text{and} \quad s_{k+1} = s_0 - \sum_{i=1}^{k+1} |\alpha_{j_i}| \leq 0. \tag{16}$$

If for this k relation $k > \ell$ holds then, by (15), we have $|t_k| > 0$. Hence, despite the presence of degeneracy, a positive step can be made towards optimality. If $k \leq \ell$ then the step will be degenerate.

Note, s_0 is the amount of bound violation of the selected primal basic variable. Therefore, (16) means that if the bound violation is greater than the sum of the $|\alpha_{j_i}|$ values in the participating degenerate positions then, despite degeneracy, the algorithm makes a positive step.

6.4 Implementation

For any dual simplex algorithm based on the revised simplex method it is important to access the original sparse matrix $\mathbf{A}$ both rowwise and columnwise. Therefore, for fast access, $\mathbf{A}$ should be stored in both ways.

For the efficient implementation of BSD a sophisticated data structure (e.g., a priority queue) is needed to store and (partially) sort the breakpoints. This part of the implementation must be done with great care because experience shows rather erratic behavior of the algorithm regarding the generated and used breakpoints. For example,

it can happen that several thousands of breakpoints are defined in consecutive iterations but in one of them only less then 10 are used while in the next one nearly all. Even the number of generated breakpoints can change by orders of magnitude up and down from one iteration to the next. The tendency is that along the simplex tail these numbers go down to moderate values and often the first one is chosen which corresponds the step of the traditional dual method.

6.5 Features

The introduced BSD algorithm possesses several advantageous features that make it the algorithm of choice for the BB type solution of (M)IP problems.

1. The more type-1 variables are present the more effective the method is. Namely, in such cases the chances of generating many breakpoints is high which can lead to larger steps towards dual optimality.

2. BSD can cope with dual degeneracy more efficiently than the traditional method as it can bypass zero valued breakpoints and thus make a positive step.

3. BSD has a better numerical behavior due to the increased flexibility in choosing the pivot element. Namely, if the maximum of $f(t)$ is defined by a small pivot element and this is not the first breakpoint then we can take the neighboring (second neighbor, etc.) breakpoint that is defined by an acceptable pivot.

4. In BSD, the incoming variable enters the basis at feasible level.

5. The computational viability of BSD lies in the multiple use of the expensively computed pivot row.

In short, the main advantage of the BSD algorithm is that it can make many iterations with one updated pivot row and these iterations cost hardly more than one traditional iteration. Such situations frequently occur when LP relaxations of MILP problems are solved.

On the other hand, BSD incorporates the usual dual algorithm and makes steps according to it when bound swap cannot be made. Consequently, it can be said that BSD is an efficient generalization of the traditional dual simplex algorithm.

7 Two examples for the algorithmic step of BSD

The operation of BSD is demonstrated on two examples. We assume the nonbasic variables are located in the leftmost positions and the nonbasic components of the updated pivot row have been determined. The types of the variables and the d_j values are given. The status of upper bounded variables is also indicated (LB for 'at lower bound' and UB for 'at upper bound').

Example-1. The problem has 10 nonbasic variables. Assume a type-2 basic variable $x_{Bp} = -11$ has been chosen to leave the basis, $f_0 = 1$. Note, the solution is dual degenerate.

j	1	2	3	4	5	6	7	8	9	10
Type(x_j)	0	1	1	1	1	1	1	1	1	2
Status		LB	LB	UB	UB	LB	UB	UB	LB	
Upper bound	0	1	1	1	1	1	2	1	5	∞
α_j^p	2	-2	1	3	-4	-1	1	-2	-1	-2
d_j	-1	2	5	-6	-2	0	0	0	4	10
Ratio		-1		-2		0	0		-4	-5

6 ratios have been defined, $Q = 6$. After sorting the absolute values of the ratios:

k	1	2	3	4	5	6
j_k	6	7	2	4	9	10
$\lvert t_k \rvert$	0	0	1	2	4	5
$\alpha_{j_k}^p$	-1	1	-2	3	-1	-2

According to the definition of the slope, $s_0 = -x_{Bp} = 11$. Now, applying Step 4 of BSD, we obtain

k	j_k	$\lvert t_k\rvert$	$\alpha^p_{j_k}$	u_{j_k}	$\theta^{(p)}$	$s_k = s_{k-1} - \lvert\alpha^p_{j_k}\rvert u_{j_k}$	x^k_{Bp}	f_k	Remarks
1	6	0	-1	1	$(-11)/(-1) = 11$	$11 - \lvert-1\rvert \times 1 = 10$	-10	1	BSW $\uparrow x_6$
2	7	0	1	2	$(-10)/(1) = -10$	$10 - \lvert-1\rvert \times 2 = 8$	-8	1	BSW $\downarrow x_7$
3	2	1	-2	1	$(-8)/(-2) = 4$	$8 - \lvert-2\rvert \times 1 = 6$	-6	9	BSW $\uparrow x_2$
4	4	2	3	1	$(-6)/(3) = -2$	$6 - \lvert3\rvert \times 1 = 3$	-3	15	BSW $\downarrow x_4$
5	9	4	-1	5	$(-3)/(-1) = 3$	$3 - \lvert-1\rvert \times 5 = -2$	0	21	

Here, x^k_{Bp} denotes the value of the selected basic variable after k sections of $f(t)$ have been passed, f_k is the value of the dual objective in breakpoint k, BSW $\uparrow$ indicates a bound swap from lower to upper bound, while BSW $\downarrow$ denotes the opposite. In the first four steps the displacement of the actual incoming variable ($\theta^{(p)}$) was always greater than its own upper bound (u_{j_k}), therefore they all signalled a bound swap and the algorithm proceeded. At the fifth breakpoint the opposite happened, therefore x_9 (that defined t_5) became the incoming variable with a value of 3 (which is less than $u_9 = 5$). The dual steplength is 4 ($= -t_5$). At the end, we used five breakpoints out of six resulting in a progress of the dual objective from 1 to 21. We have bypassed two degenerate vertices. Dual-2 would have stopped at the first breakpoint resulting in a non-improving (degenerate) iteration and a newly introduced infeasible basic variable $x_6 = 11 > 1 = u_6$.

It is worth looking at the situation after the BSD iteration.

j	1	2	3	4	5	6	7	8	9	10
Type(x_j)	0	1	1	1	1	1	1	1	1	2
Status		UB^*	LB	LB^*	UB	UB^*	LB^*	UB	B^*	
$\bar{d}_j$	7	-6	9	6	-18	-4	4	-8	0	2

Here, '*' denotes a changed status in the Status row, B denotes basic. Clearly, the solution has changed quite substantially, even dual degeneracy has disappeared.

Example-2. This problem has 6 nonbasic variables. Now, a type-1 basic variable, $x_{Bp} = 14$, has been chosen to leave the basis because it is above its upper bound of 2, $f_0 = 6$. Again, the solution is dual degenerate.

j	1	2	3	4	5	6
Type(x_j)	1	1	1	1	1	2
Status	LB	LB	UB	UB	LB	
Upper bound	5	1	1	2	2	∞
α_j^p	-2	3	-2	-1	1	3
d_j	2	3	0	-2	0	9
Ratio		1	0	2	0	3

5 ratios have been defined, $Q = 5$. The ratios are positive now. For uniformity, we still refer to their absolute value. After sorting them:

k	1	2	3	4	5
j_k	3	5	2	4	6
$\|t_k\|$	0	0	1	2	3
$\alpha_{j_k}^p$	-2	1	3	-1	3

Now the slope is defined as $s_0 = x_{Bp} - u_{Bp} = 14 - 2 = 12$. Applying Step 4 of BSD, we obtain

k	j_k	$\|t_k\|$	$\alpha_{j_k}^p$	u_{j_k}	$\theta^{(p)}$	$s_k = s_{k-1} - \|\alpha_{j_k}^p\| u_{j_k}$	f_k	Remarks
1	3	0	-2	1	$12/(-2) = -6$	$12 - \|-2\| \times 1 = 10$	6	BSW $\downarrow x_3$
2	5	0	1	2	$10/(1) = 10$	$10 - \|1\| \times 2 = 8$	6	BSW $\uparrow x_5$
3	2	1	3	1	8/3	$8 - \|3\| \times 1 = 5$	14	BSW $\uparrow x_2$
4	4	2	-1	2	$5/(-1) = -5$	$5 - \|-1\| \times 2 = 3$	19	BSW $\downarrow x_4$
5	6	3	2	∞	$3/(3) = 1$	$-\infty$	22	

As expected the BSD iteration has made a major reorganization of the solution. The following table shows the situation after the BSD iteration has been completed. Dual degeneracy has, again, disappeared. For notations, see Example-1.

j	1	2	3	4	5	6
Type(x_j)	1	1	1	1	1	2
Status	LB	UB^*	LB^*	LB^*	UB^*	B^*
$\bar{d}_j$	8	-6	6	1	-3	0

The above examples suggest that in case of large scale problems with many upper bounded variables the favorable features of the BSD algorithm have good chances to materialize.

8 Computational experience

Though the paper is not meant to be a computational study we still find it appropriate to give some indication of the performance of BSD on real life problems.

As BSD is a dual phase-2 algorithm we have selected problems for which the all-logical basis is dual feasible or can be made so by swapping the bound status of bounded nonbasic variables. Four problems (`fit2d, fit2p, gfrd-pnc, nesm`) are from the `netlib/lp` library of LP problems and another four (`air05, mitre, mod011, mkc`) are from the `MIPLIB 3.0` library of mixed integer programming (MILP) problems. In the latter case we considered the solution of the LP relaxation of the MILP problems.

				Number of variables by type			
Name	Rows	Columns	Nonzeros	type-0	type-1	type-2	type-3
`fit2d`	26	10500	138018	0	10500	0	0
`fit2p`	3001	13525	60784	0	7500	6025	0
`gfrd-pnc`	617	1092	3467	0	258	834	0
`nesm`	663	2923	13988	175	1508	1240	0
`air05`	426	7195	59316	0	7195	0	0
`mitre`	2054	10724	49028	0	10724	0	0
`mkc`	3411	5325	19984	0	5323	2	0
`mod011`	4480	10958	37425	1	1596	9361	0

Depending on the type of constraints, the added logical variables can also be of any type.

We were interested in the total number of iterations made by BSD as compared to Dual-3 (which always takes $k = 1$). In this way, we believe, a first idea can be obtained about the overall effectiveness of allowing multiple steps (BSD) versus single step (Dual-3). The table below gives the number of iterations to optimality. Tests were performed with no presolve, starting from the all-logical basis and using Dantzig pricing (selecting row with largest violation). This is by no means the most ideal strategy but it enables a possible comparison with other algorithms.

Name	BSD itns general k	Dual-3 itns $k = 1$
fit2d	184	7096
fit2p	13918	34878
gfrd-pnc	553	589
nesm	1971	4171
air05	3851	9571
mitre	2144	11975
mkc	433	3378
mod011	5911	5784

Few comments on the results are the following.

Name	BSD iterations with $k > 1$	Dual-3
`fit2d`	Most of them.	Many degenerate iterations.
`fit2p`	Many	Many degenerate iterations.
`gfrd-pnc`	Very few.	Hardly any degenerate iterations.
`nesm`	More than 30%.	Several short (up to 10 iterations long) rallies of degenerate iterations.
`air05`	More than 80%.	Several short rallies of degenerate iterations.
`mitre`	Many in the first 600 iterations, very few afterwards.	Many degenerate iterations.
`mkc`	Very many until iteration 160. In particular, bypassed 2793 break points in iteration 2. Very few after iteration 160.	Many degenerate iterations.
`mod011`	Hardly any.	Many short rallies of degenerate iterations.

These results suggest that if a larger proportion of variables are bounded and the problem is degenerate to some extent then BSD can make substantial gains in the number of iterations. On the other hand, there is no theoretical guarantee that the bound swapping iterations (if there are any) reduce the total number of iterations (c.f. `mod011`). To draw some more conclusions (or even to confirm the present ones) a detailed computational analysis need to be performed.

9 Summary

We have presented a generalization of the dual phase-2 algorithms that has the potential to handle all types of variables efficiently. It is based on the piecewise linear nature of the dual objective if defined as a function of relaxing one basic dual equation. The distinguishing features of this method are: (i) in one iteration it can make progress of many traditional

dual iterations, (ii) using proper data structures it can be implemented very efficiently so that an iteration requires hardly more work than the traditional pivot method, (iii) it has inherently better numerical stability because it can create a large flexibility in finding a pivot element, (iv) it can be very effective in coping with degeneracy as it can bypass dual degenerate vertices more easily than the traditional pivot procedures. The operation of the method is demonstrated through examples and some limited computational experience is also presented.

10 Acknowledgements

The work presented in this paper has benefited from fruitful discussions with Tamás Terlaky for which the author expresses his gratitude. Support and encouragements from Gautam Mitra are also acknowledged.

The helpful remarks of two anonymous referees which the author gratefully acknowledges have contributed to the improvement of the paper.

References

[1] Bixby, R.E., Fenlon, M., Gu, Z., Rothberg, E., Wunderling, R., "MIP: Theory and practice—closing the gap", in Powell, M.J.D., Scholtes, S., (eds.), *System Modelling and Optimization: Methods, Theory and Applications*, Kluwer (2000), pp. 19–50,

[2] Chvátal, V., *Linear Programming*, Freeman and Co., 1983.

[3] Dantzig, G.B., "Maximization of a linear function of variables subject to linear inequalities", in Koopmans, T.C. (ed.), *Activity analysis of production and allocation*, John Wiley & Sons, New York, 1951, p. 339–347.

[4] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.

[5] Fourer, R., "Notes on the Dual Simplex Method", Unpublished, March, 1994.

[6] Greenberg, H.J., "Pivot selection tactics", in Greenberg, H.J. (ed.), *Design and Implementation of Optimization Software*, Sijthoff and Nordhoff, 1978.

[7] Lemke, C.E., "The Dual Method of Solving the Linear Programming Problem", *Naval Research Logistics Quarterly*, 1, 1954, p. 36–47.

[8] Maros, I., "A general Phase-I method in linear programming", *European Journal of Operational Research*, 23(1986), p. 64–77.

[9] Maros, I., "A Piecewise Linear Dual Procedure in Mixed Integer Programming", in F. Giannesi, R. Schaible and S. Komlosi (eds.), *New Trends in Mathematical Programming*, Kluwer Academic Publishers, 1998, p. 159–170.

[10] Maros, I., Mitra, G., "Simplex Algorithms", Chapter 1 in Beasley J. (ed.) *Advances in Linear and Integer Programming*, Oxford University Press 1996.

[11] Nemhauser, G.L., Wolsey, L.A., *Integer and Combinatorial Optimization*, John Wiley, 1988.

[12] Orchard-Hays, W., *Advanced Linear-Programming Computing Techniques*, McGraw-Hill, 1968.

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

[14] Wolfe, Ph., "The composite simplex algorithm", *SIAM Review*, 7 (1), 1965, p. 42–54.