Pivot Rules for Linear Programming: A Survey on Recent Theoretical Developments

T. Terlaky and S. Zhang

November 1991

Address of the authors:

Tamás Terlaky [1]
Faculty of Technical Mathematics and Computer Science,
Delft University of Technology,
P.O. Box 5031,
2600 GA Delft, The Netherlands.

Shuzhong Zhang
Department of Econometrics,
University of Groningen,
P.O. Box 800,
9700 AV Groningen, The Netherlands.

[1]On leave from the Eötvös University, Budapest, and partially supported by OTKA No. 2115.

Abstract

The purpose of this paper is to discuss the various pivot rules of the simplex method and its variants that have been developed in the last two decades, starting from the appearance of the minimal index rule of Bland. We are mainly concerned with finiteness properties of simplex type pivot rules. Well known classical results concerning the simplex method are not considered in this survey, but the connection between the new pivot methods and the classical ones are discussed if there is any.

In this paper we discuss three classes of recently developed pivot rules for linear programming. The first and largest class is the class of essentially combinatorial pivot rules including minimal index type rules and recursive rules. These rules only use labeling and signs of the variables. The second class contains those pivot rules which can actually be considered as variants or generalizations or specializations of Lemke's method, and so they are closely related to parametric programming. The last class has the common feature that the rules all have close connections to certain interior point methods. Finally, we mention some open problems for future research.

Key Words: Linear programming, simplex method, pivot rules, cycling, recursion, minimal index rule, parametric programming.

1 Introduction

In this paper, we consider the following linear programming problems in standard form:

$$(P) \qquad \min\{c^T x : Ax = b,\ x \geq 0\},$$

$$(D) \qquad \max\{b^T y : A^T y \leq c\},$$

where A is an $m \times n$ matrix, and b, c, x, y are m and n dimensional vectors respectively. The problem (P) is called the *primal* problem and (D) the *dual* problem.

The main purpose of this paper is to give an overview of the various pivot rules for solving linear programming problems either in the form (P) or in the form (D). A pivot method is called a *simplex method* if it preserves the (primal or dual) feasibility of the basic solution.

Linear programming has been one of the most dynamic areas of applied mathematics in the last forty years. Although recently many new approaches to solve a linear programming problem have been suggested, Dantzig's (cf. [20]) simplex method (for an implementation see e.g. [5, 23]) still seems to be the most efficient algorithm for a great majority of practical problems. Solving these problems, it turns out that in most cases the required number of pivot steps is a linear (at most quadratic) function of the problem dimension. This observation about the practical efficiency of the simplex method [36] is theoretically supported by proving its polynomial behavior in the expected number of pivot steps [11, 2, 85]. On the other hand, for most of the simplex variants there are examples for which the number of pivot steps are exponential [3, 49], [60, 61, 67] and [72]. To find a *polynomial pivot rule* for linear programming, i.e. a pivot rule which bounds the number of pivot steps by a polynomial function of the number of variables, or to prove that such a pivot rule does not exist, seems to be very hard. Actually, the so-called "Hirsch conjecture" [50] and to clarify whether there is such a polynomial time simplex algorithm are the most challenging open problems in polyhedral theory and linear programming.

One reason for the efficiency of the simplex method even in the case of nondegeneracy may be that the simplex method is very flexible, i.e. there are a lot of possibilities to select the pivot element and to avoid immediate numerical difficulties. Hence it is not surprising that many simplex variants have been developed.

However, in recent years most attention and research in linear programming have been devoted to new methods like the ellipsoid method developed by Khachian [44] and later on (even more intensively) to the interior point methods initiated by Karmarkar [43]. Recent papers concerning simplex pivot rules have not been receiving much attention, even among researchers in the field of linear programming. Moreover, a very large part of this research was presented in terms of oriented matroid programming and frequently not specialized to pivot rules for linear programming. Also some of the other results were obtained as a side result (extreme case) of some interior point methods. Due to this a lot of results remained unknown to researchers only working on the simplex method. Our aim here is to summarize the pivot rules that have appeared in the last two decades. Since most of the early classical results concerning simplex pivot rules are well known, we shall start

our discussion from the middle seventies. The earlier results are mentioned only when it is necessary to relate them to some new algorithms, or to assist the understanding. Note that there are rich research results concerning pivot rules for specially structured linear programming problems, like network linear programming, assignment problems, etc. Due to their independent interest, we will not include these results in this survey.

The paper is organized as follows. At the end of this section we will present some standard notations which will be used throughout the paper. In Section 2 we will discuss simplex type algorithms that are related to or presented exclusively in oriented matroid programming. Due to their combinatorial character, all of these methods are finite even in the case of degeneracy. This class includes Bland's [8, 9] minimal index rule (which is related to Murty's Bard type scheme [59] and Tucker's consistent labelling [86]), Bland's recursive rule [8, 9], the Edmonds–Fukuda rule [24] and its variants [18, 92, 93, 88], Jensen's general relaxed recursion [41], the finite criss–cross method [13, 79, 80, 87] and its variants [25, 89] and finally Todd's rule [82, 83] which was presented for the linear complementarity problem of oriented matroids. All these methods, except for the criss–cross method and Jensen's recursion, are simplex variants, i.e. they preserve the feasibility of the bases in the case of LP. Bland's recursive rule is the first and general recursive scheme of the simplex method preserving the feasibility, whereas Jensen's work generalizes this recursion to infeasible bases. Many other finite pivot rules, once their implicit recursive properties are established, can be interpreted as specific implementations of these general schemes. However, Todd's rule uses the principal idea from one of the first finite methods for the simplex approach, i.e. the lexicographic method, and so it is not recursive in nature. It is interesting to note that in the case of oriented matroid programming only Todd's rule preserves feasibility of the bases and finiteness simultaneously and therefore yields a finite simplex algorithm. It is also remarkable that Bland's minimal index rule is not finite for oriented matroid programming.

Criss–cross methods gave a positive answer to the following question: Does there exist a finite (may be infeasible) pivot method for the simplex type approach which solves the LP problem in one phase? There were several attempts to relax the feasibility requirement in the simplex method. The first step was made by Dantzig [20] by presenting his parametric self dual algorithm, which can be interpreted as Lemke's [54] algorithm for the corresponding linear complementarity problem [55]. The first criss–cross type method was designed by Zionts [94]. The finite criss–cross algorithm was presented independently by Chang [13], Terlaky [79, 80] and Wang [87]. Terlaky presented this algorithm for linear and oriented matroid programming. In his unpublished report [13] Chang presented this algorithm for positive semidefinite linear complementarity problems, while Wang [87] presented it for oriented matroid programming. Criss–cross methods, like the parametric self–dual method [20], can be initiated with any basic solution, and will finally find the optimal solution or detects infeasibility. Unfortunately, the finiteness of Zionts' criss–cross method is not clear in general, while Terlaky's criss–cross method is finite. Other combinatorial type algorithms have arisen recently. Jensen's [41] general recursive scheme is actually a general frame for finite recursive pivot algorithms.

In Section 3, we present the second class of pivot rules. These pivot rules have a common feature, namely all of them can be interpreted as a parametric or iteratively reparametrized simplex algorithm. These algorithms may cycle for degenerate problems if no anti-cycling techniques are incorporated. In particular, the variants of Magnanti and Orlin's [56] parametric algorithm handle degeneracy explicitely and have the anti-cycling property. The parametric programming procedure related to these algorithms is often referred to as the shadow vertex algorithm [31, 11, 60]. This variant of the simplex method is also exponential in the worst case [34]. However, under some reasonable probabilistic assumptions its expected number of pivot steps is polynomial [11]. The algorithms reviewed in this section are the Monotonic Build-Up simplex algorithm (MBU) [1], Magnanti and Orlin's [56] parametric algorithm and the Exterior Point Simplex Algorithm (EPSA) [68, 69, 70]. The last one can also be considered as a special case of MBU. The relations of these algorithms to Dantzig's [20] self–dual parametric simplex algorithm are also discussed. At the end of Section 3, we mention the Transition Node Pivoting (TNP) algorithm [32, 30].

In Section 4, we discuss the third group of pivot rules. In this group, the simplex pivot algorithms were derived from or inspired by interior point methods. We will include Todd's algorithm [84] derived from a projective interior point method as the limit, the minimum volume simplex pivot rule of Roos [71] which is related to Karmarkar's [43] potential function, and the dual-interior-primal-simplex method of Tamura *et al.* [78] as a generalization of Murty's [62, 63] gravitational method.

For the three classes of pivot rules to be discussed in this survey, the degeneracy issue is emphasized especially for pivot rules included in Section 2. For the parametric programming oriented pivot rules presented in Section 3, only the Magnanti and Orlin algorithm [56] and the transition node pivot rule [32, 30] particularly handle the degeneracy problem. The pivot rules in Section 4 mostly need nondegeneracy assumptions to establish finiteness. This shows again the necessity and importance of handling degeneracy problems in optimization.

To conclude, some remarks and open problems will be mentioned at the end of the paper in Section 5. We also give some hints to the literature of practically implemented anti-cycling methods.

To unify the presentation, the following notations will be used in this paper. Matrices will be denoted by capital letters, vectors by lower case letters and their coordinates by the same letters with subscript (e.g. if $x \in R^n$, then $x^T = (x_1, \ldots, x_j, \ldots, x_n)$). Given a linear programming problem, a *basis* B is a maximal subset of indices $\{1, 2, ..., n\}$ such that the corresponding column vectors of the matrix A (i.e. $\{a_i : i \in B\}$) are independent. The submatrix consisting of these columns is denoted by A_B and the *basic tableau* $T(B)$ corresponding to this basis B is schematically depicted in Figure 1.

a_1	...	a_s	...	a_j	...	a_n	
		t_{is}		t_{ij}		x_i	a_i
		z_s		z_j			

Figure 1.

In Figure 1, t_{ij} represents the coefficient of the basic vector a_i in the basic representation of the vector a_j. Note that in our interpretation, the index pair ij of the tableau refers to the basic vector a_i and the nonbasic vector a_j, but not to the position of t_{ij} in the tableau. Moreover, the vector x_i denotes the i-th coordinate of the *basic solution* on the right hand side of the basic tableau $T(B)$ (cf. Figure 1). If there is no confusion x_i is also called a *basic variable* for $i \in B$. We call $N := J \setminus B$ the *nonbasis* and x_j a *nonbasic variable* for $j \in N$, where $J := \{1, 2, ..., n\}$. The row vector $y^T = c_B^T A_B^{-1}$ represents the corresponding dual solution and the variable z_j the j-th coordinate of the *reduced cost vector* z given by $z^T = (z_1, \dots z_n) = c^T - y^T A$. Finally, denote by ${t^{(r)}}^T = (t_{r1}, \dots t_{rn})$ the row in the tableau $T(B)$ corresponding to the basic index r.

We also define a vector $t_{(k)}$ for all $k \in N$ as follows:

$$t_{(k)} = (t_{(k)j})_{j=1}^n \quad \text{with} \quad t_{(k)j} = \begin{cases} t_{kj} & \text{if } j \in B, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that the vector $t_{(k)}$ is a solution of the homogeneous system $Ah = 0$.

A basis (tableau) is called *primal feasible* if $x_i \geq 0$ for all $i \in B$ and it is called *dual feasible* if $z_j \geq 0$ for all $j \in N$. If a basic variable x_i leaves the basis and a nonbasic variable x_j enters, then this operation (transforming the tableau) is called *pivoting* on position (i, j) and element t_{ij} is referred to as the *pivot element*.

If all the basic variables are nonzero and all the nonbasic variables have nonzero reduced cost, then the basis is called *nondegenerate*, else the basis is called *degenerate*. We say that the basis is *primal degenerate* if at least one basic variable is zero, and *dual degenerate* if at least one nonbasic variable has zero reduced cost.

2 Combinatorial and Finite Pivot Rules

In this section we will review those recently developed pivot rules which are concerned only with the "infeasibility" of the basic variables or the nonbasic variables with negative reduced costs (dual infeasible). We call these type of pivot rules *combinatorial pivot rules*. In other words, combinatorial pivot rules take only care of the signs of the elements in the last row and the last column of the tableau. An example of a noncombinatorial pivot rule is given by the well known Dantzig's pivot rule which always selects the nonbasic variable with the least (negative) reduced cost, and so Dantzig's rule concerns not only with the signs of the variables but also with their magnitudes. Such rules will not be discussed in this section.

In fact, most combinatorial pivot rules are originally developed in the context of an abstracted form of linear programming, i.e. oriented matroid programming. The theory of oriented matroid programming was established by Bland [9], Folkman and Lawrence [22] and Fukuda [24]. In this section we will mainly focus on the presentation of these results in a linear programming terminology.

We start our discussion with the well known pivot rules of Bland [8]. Actually, in Bland's paper [8], two pivot methods are proposed. Those rules start from a feasible basis and keep the primal feasibility. Bland's first rule is known as the "minimal index rule", which due to its elegant simplicity received much attention in the last decade. The minimal index rule is as follows:

We start with a primal feasible tableau $T(B)$, and fix an arbitrary ordering (indices) of the variables.

Bland's Minimal Index Rule

Step 1 (Entering variable selection) Choose the nonbasic variable x_s whose index is minimal among the variables with negative reduced cost.

If there is no such variable, $T(B)$ is optimal, stop.

Step 2 (Leaving variable selection) Choose a variable, say x_r, to leave the basis such that after pivoting on position (r, s) the basis remains feasible and index r is the minimal among all indices with this possibility.

If such a variable does not exist, the problem is unbounded, stop.

Remark that the procedure to maintain the feasibility of the basis is normally called the *minimum-ratio test* in the simplex method.

Bland's second rule is less well known. In fact, Bland's second rule proposes an *inductive* (or *recursive*) procedure. The idea of that procedure is based on the elementary observation that a primal feasible basis would be optimal if the dual infeasible variables would be deleted. Bland's recursive rule can be described as follows.

We start with a primal feasible tableau $T(B)$.

Bland's Recursive Rule

Step 0 Fix all dual infeasible variables to zero level, therefore a subproblem is solved optimally.

Step 1 To get a subproblem with size one larger release one fixed variable (with negative reduced cost), preserving primal feasibility pivot this variable in, and fix it as a basis variable (free variable, or equivalently the corresponding dual variable is fixed to zero) . The number of variables in this subsidiary problem is again the same as we have solved previously. Solve this subsidiary problem recursively.

If it is unbounded, then the original problem is unbounded.

Step 2 If we have an optimal solution of the subproblem, release all fixed variables.

If there is no dual infeasible variable then the actual solution is optimal. Else repeat the procedure.

Remark. A *recursive pivot rule* (in the primal form) introduces a new variable entering the basis if all the previously introduced variables are in a feasible basis or have nonnegative reduced costs.

By the lemma provided below we shall see that for successive degenerate pivot steps, the minimal index rule works on the simplex tableau in a recursive way. We will further show that a large class of combinatorial pivot rules are recursive for degenerate pivots and therefore finite. We note here that Murty's Bard type scheme [59] and Tucker's consistent labelling algorithm [86] (for maximum flow problem) were proposed independently based on similar ideas. First we will present the following important lemma for proving the finiteness of most anti-cycling pivot rules.

Lemma 1 *Let $T(B)$ and $T(B')$ be two different tableaus. If for these tableaus $T(B)$ and $T(B')$ the vectors $t_{(k)}$ for any $k \notin B$ and $t^{(r)}$ for any $r \in B'$ are considered, it follows that*

$$\begin{aligned}(z - z')^T(x - x') &= 0,\\ (z - z')^T t_{(k)} &= 0,\\ (x - x')^T t^{(r)} &= 0,\\ t_{(k)}^T t^{(r)} &= 0.\end{aligned}$$

Proof:
Cf. [8, 9, 79, 80]. □

Observe by extending the vectors x, z, $t_{(k)}$ and $t^{(r)}$ with 0-th and $(n+1)$-th coordinates given by $x_0 = z_{n+1} =$ the objective value; $x_{n+1} = -1$; $z_0 = 1$; $t_{(k)n+1} = t_0^{(r)} = 0$; $t_{(k)0} = z_k$ and $t_{n+1}^{(r)} = x_r$ and denoting the extended vectors by $\overline{x}, \overline{z}, \overline{t}_{(k)}^T$ and $\overline{t}^{(r)}$ that the relations

$$\overline{x}^T \overline{z}' = 0, \;\; \overline{t}_{(k)}^T \overline{z}' = 0, \;\; \overline{x}^T \overline{t}^{(r)} = 0, \;\; \overline{t}_{(k)}^T \overline{t}^{(r)} = 0$$

hold.

Using Lemma 1 we are able to show the following fundamental theorem.

Theorem 1 *For any simplex pivot rule (in the primal form) satisfying one of the following two properties:*

1) *(*nondegeneracy*) either a pivot step is nondegenerate; or*

2) *(*recursive*) any successive pivot steps are recursive,*

then the pivot rule is finite.

Proof:
We need only to prove that no cycling will occur in degenerate pivot steps. Suppose to the contrary, that cycling occurs. We delete the columns and the rows corresponding to those variables that do not change their status from basic to nonbasic or the other way round in the cycle. It is clear that the cycle remains in the reduced tableaus.

Now, let x_k denote the last involved nonbasic variable in the cycle that becomes basic, and the corresponding tableau in which x_k is entering the basis be T_1. Since the pivot steps are recursive, we conclude that all the other nonbasic variables in T_1 have nonnegative reduced costs. By the definition of a recursive procedure, we should first solve after that pivot step the subproblem obtained by deleting the row and the column corresponding to the variable x_k. If the subproblem is solved optimally, then with x_k a degenerate basic variable the problem is also solved optimally. This contradicts the fact that a cycle exists. On the other hand, if the subproblem is unbounded whereas by adding the basic variable x_k it is not, it follows that in the last tableau the column of the entering basis variable x_r has only one positive element in the cross of the row corresponding to x_k. Taking this column and by Lemma 1, the corresponding vector $\overline{t}_{(r)}$ is orthogonal to the last row $\overline{z}'$ of the tableau T_1. However, for these two vectors each coordinate does not have the same sign. In particular, the coordinate corresponding to the variable x_k has opposite signs in $\overline{t}_{(k)}$ and in $\overline{z}'$. This again results in a contradiction (in this case $\overline{t}_{(k)}^T \overline{z}'$ is negative instead of zero) to Lemma 1 and so the theorem is proved.

□

It is interesting to see that the following pivot rules satisfy the properties discussed in the above theorem.

1. Bland's minimal index rule;

2. The Last In First Out rule, i.e. select the nonbasic variable which has negative reduced cost and which has become nonbasic most recently, and select the variable to leave the basis according to the minimum-ratio test and if there are more than one candidates select the one which has become basic most recently;

3. The Most Often Selected rule, i.e. select variables to enter and to leave the basis with the most often selected historical record.

These rules were discussed in Zhang [92]. Moreover, the so-called *well-preserved ordering* property discussed in Zhang's paper is essentially the same as the recursive property in Theorem 1.

Interestingly, in the combinatorial abstraction of linear programming — oriented matroid programming, *nondegenerate cycling* can take place. Examples of this type can be found in Fukuda [24] and Ziegler [93]. Note that the minimal index pivot rule is recursive only for degenerate pivot steps. We may call this type of rules *partial recursive*. So the partial recursive rules may still cycle in the oriented matroid programming case. Attention was paid to find finite pivot rules in that case. Bland's general recursive rule satisfies this requirement. We shall remark here that Bland's recursive rule ([8]) is very general. It is actually a general scheme which gives many flexibilities and guarantees finiteness within the framework. However, a naive implementation of this scheme may lead to the difficulty that many historical records will be required as "bookkeeping" in the procedure. The well known minimal index rule of Bland is an implicit partial recursive method. But it is elegant and simple for itself. No explicit recursive recording is needed in that rule, and it is easily understood. The rule was independently presented together with the general scheme in [8]. Edmonds and Fukuda suggested a simple pivot rule (cf. [24]) which they proved to be finite even for oriented matroid programming. It turned out later that this rule is also implicitly recursive and therefore can be interpreted as a specific implementation of Bland's recursive scheme. Again, in Edmonds–Fukuda's rule no explicit "bookkeeping" is needed. Moreover, some flexibilities remain. We shall now introduce this rule. We also wish to show this way how interesting special algorithms, which have their own nice properties (e.g. the way to use recursion implicitly), can be fitted in general schemes. This might inspire further research to find simple, easily implementable recursive pivot rules based on the merits of Bland's general recursion.

We state Edmonds–Fukuda's rule here in terms of linear programming. Remark that Edmonds–Fukuda's rule is in fact recursive comparing to the partial recursive ones discussed in Theorem 1, although its presentation is not recursive.

We start with a primal feasible tableau $T(B)$.

Edmonds–Fukuda Rule

Step 0 From $T(B)$ we construct a list $L := (l_1, ..., l_t)$ where $\{l_1, ..., l_t\}$ is the index set of nonbasic variables in T_B.

Step 1 (Entering variable selection) Choose the nonbasic variable x_{l_k} with negative reduced cost whose index is the rightmost in L.

If there is no such variable, $T(B)$ is optimal, stop.

Step 2 (Leaving variable selection) Choose a variable, say x_r, to leave the basis such that after this pivot operation the basic variables in the set $\{x_i : i \in B \setminus \{l_1, ..., l_k\}\}$ remain primal feasible. (Partial minimum–ratio test)

If such a variable does not exist, the problem is unbounded.

Step 3 (Update) Pivot on (r, l_k) and let

$B := B \setminus \{r\} \cup \{l_k\}$,

$N := J \setminus B$,

$L := (l_1, ..., l_k, l'_{k+1}, ..., l'_s)$

where $\{l'_{k+1}, ..., l'_s\} := \{l_{k+1}, ..., l_t, r\} \cap N$, (here this subset can be reordered)

$t := s$, $l_j := l'_j$, $j = k+1, ..., t$.

Go to **Step 1**.

The finiteness proof of this pivot rule in linear programming case is similar to Theorem 1. We will leave it to the reader.

An interesting question arises here. Since the primal "feasibility" is only partially maintained in the above algorithm, can we say that all tableaus produced are feasible? The answer is negative for the oriented matroid programming case (see Jensen [41] and Ziegler [93]). However, for linear programming, the Edmonds–Fukuda rule preserves primal feasibility and this can easily be seen from its geometrical interpretation. Note that Clausen [18] presented an algebraic proof for the feasibility of a modified (restricted) version of Edmonds–Fukuda rule.

Concerning recursive type pivot rules, another interesting and extremely simple pivot rule is the so-called *finite criss–cross* rule. The criss–cross method was developed independently by Chang [13] (for linear complementarity problems), by Terlaky [79, 80] (for linear and oriented matroid programming) and by Wang [87] (for oriented matroid programming). The finite criss–cross rule is based on the following two observations. First, since one is only interested in finding the optimal tableau (or basis) of a linear program, it is not always necessary to maintain either primal or dual feasibilities as the primal or the dual simplex method does. Hopefully there can even be some "short-cut" if we allow primal and dual infeasibilities. Second, the recursive scheme can be carried out more

thoroughly if we do not insist on maintaining the primal feasibility (in the primal simplex form) during the pivot iterations.

Originally a criss–cross method was proposed by Zionts [94] for linear programming. However, there is no guarantee of finiteness in Zionts' criss–cross method. The finite criss–cross method [79] was developed based on the minimal–index recursive technique. The method is surprisingly simple and requires neither primal nor dual feasible basis to start with. The method also remains finite for oriented matroid programming (Terlaky [80] and Wang [87]) and can be presented as follows.

Finite Criss–cross Rule

Step 0 Let $T(B)$ be an arbitrary tableau (can be neither primal nor dual feasible);

Step 1 Let

$r := \min\{i : \text{ either } x_i \text{ is primal infeasible or } x_i \text{ has a negative reduced cost}\}$;

If there is no r, then $T(B)$ is optimal, stop.

Step 2 If x_r is primal infeasible, let $s := \min\{l : t_{rl} < 0\}$.

Go to **Step 3.1** (if there is no s, the problem is infeasible, stop).

If x_r has a negative reduced cost, let $s := \min\{l : t_{lr} > 0\}$.

Go to **Step 3.2** (if there is no s, the problem is dual infeasible, stop);

Step 3.1 Pivot on (r, s). Go to **Step 1**.

Step 3.2 Pivot on (s, r). Go to **Step 1**.

The finiteness proof of this pivot rule (in linear programming case) is similar to the proof of Theorem 1. For details, one is referred to Terlaky [79].

Note that the finite criss–cross method presented above is completely combinatorial. An exponential example for the criss–cross pivot rule is given by Roos [72]. The construction of that example is closely related to the Avis-Chvátal example [3] for the minimal index pivot rule. A new and interesting finiteness proof for the criss–cross method is given by Fukuda and Matsui [25]. Fukuda and Matsui pointed out that there can be more flexibilities in the criss–cross method without loosing finiteness. We will come back to this point later.

We remark here that these combinatorial pivot rules can be used to solve linear complementarity problems as well. For papers discussing pivot rules for the complementarity problem, among others, cf. Chang and Cottle [14], Fukuda and Terlaky [27, 28], Terlaky [81], Chang [13], Klafszky and Terlaky [47, 48], Fukuda and Namiki [26], Cottle [19] and den Hertog, Roos and Terlaky [39]. Particularly, a quadratic programming problem can be converted to a linear complementarity problem.

Since minimal index type methods need the indices of variables as an indication of priorities of subproblems to be solved recursively, it is profitable to find a "good" ordering of indices before hand. We can treat such an ordering procedure as *pre-conditioning*. Pan [66] observed that a facet of a polytope is likely to contain the optimal vertex if the angle between its tangent direction and the objective direction is small. Based on this observation, he suggested an ordering of variable indices according to the cosine of the constraints' tangent direction and the objective direction. In fact, Künzi and Tzschach already took advantage of this observation in 1965 when presenting their Duoplex-Algorithm (cf. [53]).

Now we consider again the criss–cross method. We notice that the criss–cross method is a *completely* recursive procedure, and so it can be linked to the Edmonds–Fukuda rule which is also completely recursive. This fact is observed by Fukuda and Matsui [25]. They remarked that in order to guarantee the finiteness in the criss–cross rule, only a partial ordering of variables is sufficient. At each pivot step we select a variable which is either primal infeasible or has a negative reduced cost according to this partial ordering (from low to high), and then we select the pivot element again according to this partial ordering (refer to the criss–cross rule presented above). Denote x_t to be the higher one according to the partial ordering among entering and leaving variables. In the next iteration, for those variables higher than x_t we keep the same ordering, and for those variables lower than x_t we let them equal and lower than x_t in the new partial ordering. This gives more flexibility in the criss–cross method and finiteness still can be proved. In fact Fukuda and Matsui [25] and Namiki and Matsui [65] explored more flexibility of the criss–cross method.

Based on the ideas of Bland's recursive algorithms, Jensen [41] constructed a general framework of recursive pivot rules for oriented matroid programming. To ease the presentation of Jensen's rule the following convention is used. A tableau for an LP will be called *terminal* if it shows either primal infeasibility (a negative basic variable with no negative element in the corresponding row), dual infeasibility (a dual infeasible variable with no positive element in the corresponding column) or it is optimal. A subproblem SLP will refer to a subproblem of the LP problem where a certain set of nonbasis variables are fixed as zero nonbasis variables and another subset of basis variables is fixed as nonrestricted basis variables. In solving a subproblem SLP we allow pivots where only non fixed variables may enter and leave the basis. Jensen's [41] scheme is as follows.

Jensen's General Recursion

Step 0 Let $T(B)$ be an arbitrary tableau (can be neither primal nor dual feasible);

Step 1 Select a subproblem SLP (subtableau) for which the present tableau is terminal.

Step 2 Select a variable (say x_s) not included yet in the subproblem SLP and add it to this subproblem. Denote this new subproblem by SLPs (The variable might be in or out of the basis.)

If there is no such variable, the problem is solved, stop.

Step 3.1 If the present tableau is terminal for the new subproblem SLPs, then let SLP:=SLPs and Go to **Step 2**.

Step 3.2 If the tableau is not terminal for SLPs, then make a pivot, change the position of the new variable x_s and fix it in the new position.

Step 4 The subproblem SLP' obtained so (discarding again x_s) has the same number of variables as it was solved in Step 1. Call recursively this algorithm to solve this problem. The terminal tableau for SLP' is also terminal for SLPs. Let SLP:=SLPs and Go to **Step 2**.

The finiteness and validity of Jensen's general recursion can be proved the same way as the finiteness of the criss–cross method.

Jensen's work is very general indeed. Many recursive type pivot rules, also the rules that we discussed earlier, can be viewed as special variants of Jensen's general recursive scheme. Therefore the finiteness of these pivot rules can follow from that of Jensen's scheme. Comparing to Bland's recursive rule in which the primal feasibility is preserved, Jensen's rule does not require any primal or dual feasibility. We should observe that the recursive order for the basic and nonbasic variables can be independent. For instance, it is pointed out in [92] that in the primal simplex method for linear programming the minimal index rule for selecting the entering basis variable together with the last-in-first-out rule for selecting the leaving basis variable will guarantee no cycling.

Observe also that Jensen's general recursion is a framework which does not require a simplex procedure. Any method can be applied to find the solution of the subproblems.

Up to now, we have discussed recursive type combinatorial pivot rules for linear programming. To conclude this section, we will finally discuss Todd's finite and feasible pivot rule [83]. So far we can see that most finite pivot rules are of a recursive type. These rules either do not keep feasibility or may cause *nondegenerate* cycling in oriented matroid programming. Therefore it is theoretically interesting to look for a pivot rule which is finite and keeps feasibility for oriented matroid programming. To the best knowledge of the authors, Todd's rule is the only pivot rule so far proved to be in that category. We notice here that there is a classical technique for preventing cycling in the simplex method, known as the lexicographic method. The lexicographic method has an advantage that only the selection of the leaving (basis) variable is affected by the method and yet the finiteness is guaranteed. The lexicographic method can also be interpreted as a parametric perturbation method.

The equivalence of the self–dual parametric algorithm and Lemke's method applied to the linear complementarity problem defined by the primal and dual LP problem was proved by Lustig [55]. Todd's pivot rule is related to the lexicographic Lemke method (or the parametric perturbation method), and hence using the equivalence mentioned above a simplex algorithm for LP can be derived. However, it is much more complicated to

present this method in a linear programming context. The theoretical meaning mainly exists in oriented matroid programming.

As Todd's rule is in fact a lexicographic Lemke method specialized to LP, and the perturbation is done first in the right hand side and then in the objective (with increasing order of the perturbation parameter), it gives finally a two phase simplex method. We do not present here the method completely, but only the second phase – without proofs – to illustrate its behaviour. For a complete description we refer to [83, 93].

Todd's Lexicographic Lemke Rule (Phase II)

Suppose that we are given a feasible tableau $T(B)$.

For a given set of variable indices, say $\{1, 2, ..., m\}$, the tableau is called *lexico-feasible* if

$$(x_i, t_m^{(i)}, t_{m-1}^{(i)}, ..., t_1^{(i)})$$

is lexico-positive (the first nonzero element is positive) for every $i \in B$. It is obvious that if $B = \{1, 2, ..., m\}$ then the initial tableau is lexico-feasible.

In the following, at each iteration we always select the entering variable x_r such that $z_r < 0$ and

$$(\frac{t_{(r)m+1}}{z_r}, ..., \frac{t_{(r)n}}{z_r}) = lexico \min_{z_j<0}(\frac{t_{(j)m+1}}{z_j}, ..., \frac{t_{(j)n}}{z_j}),$$

where z is the vector of the reduced costs.

For selecting the leaving variable, let the index s of the leaving variable be such that

$$(\frac{x_s}{t_{rs}}, \frac{t_m^{(s)}}{t_{rs}}, ..., \frac{t_1^{(s)}}{t_{rs}}) = lexico \min_{t_{ri}>0}(\frac{x_i}{t_{ri}}, \frac{t_m^{(i)}}{t_{ri}}, ..., \frac{t_1^{(i)}}{t_{ri}}).$$

It can be seen if $l, l+1, ..., n$ $(l \geq m+1)$ are not in the basis that the variable x_k with $k \geq l$ is selected to enter the basis only if $z_j \geq 0$ for $j \in \{1, 2, ..., m, m+1, ..., l-1\}$. Similar properties hold for variables leaving the basis. This monotonicity guarantees no cycling.

The practical behavior of this method is similar to Bland's minimal index rule. To implement Todd's rule is more complicated, but not much more expensive. A big drawback, as in general with lexicographic rules, is deciding when two entries are equal within round off errors.

It is still interesting to give an easy and clear proof of the correctness of Todd's rule. Also, it is unknown if there are some other alternative simplex pivot rules which guarantee finiteness in the context of oriented matroid programming.

This concludes this section. In the following section, we will discuss some pivot rules especially for linear programming. They may not guarantee anti-cycling if there is no special treatment. Attention is mainly paid to the numerical structure of the problem. To derive other finite variants, note that the methods discussed in the next section can be combined with (partial) recursive rules of this section and with the lexicographic rule.

3 Pivot Rules Related to Parametric Programming

The algorithms discussed in this section are closely related to parametric programming, more precisely to the shadow vertex algorithm [11, 34, 61] and to Dantzig's self–dual parametric simplex algorithm [20].

As was mentioned in the previous section, the self–dual parametric algorithm is equivalent to Lemke's method applied to the linear complementarity problem defined by the primal and dual LP problem. For details see Lustig [55]. It is also easy to see that the shadow vertex algorithm can be interpreted as a special case of the self–dual parametric simplex algorithm and so only the shadow vertex algorithm will be discussed here. Using the above mentioned correspondence the reader can easily derive the connection between the new algorithms and the self–dual parametric algorithm (and the relation to Lemke's method).

Before presenting the algorithms, note that a variable will be called a *driving variable* if it plays a special role in the pivot selection, namely the leaving (or entering) variable is subsequentially determined by using this column (or row).

Anstreicher and Terlaky [1] presented the monotonic build-up simplex algorithm (MBU). We start our discussion by presenting MBU. This is due to the fact that, as will be shown, the Exterior Point Simplex Algorithm [68, 69] (EPSA), the shadow vertex algorithm and Magnanti and Orlin's [56] parametric programming algorithm can be interpreted as special implementations of MBU.

There are two interesting properties of MBU. On one hand there are two ratio tests in every pivot step — both the leaving and entering variables are selected by performing a ratio test. Dual feasibility (for the variables that are already dual feasible) is preserved during the algorithm – due to the second ratio test – while primal feasibility is temporarily lost, but recovered time to time. On the other hand, MBU is in fact a simplex method, since a primal feasible basis is associated with each basis of MBU.

Monotonic Build Up Simplex Algorithm MBU

As in the primal simplex method, let us assume that a primal feasible basic solution is available. The pivot rule is as follows.

MBU - The Pivot Rule

Step 0 Let $J^+ = \{j : z_j \geq 0\}$, $J^- = J \setminus J^+ = \{j : z_j < 0\}$.

Step 1 Let $k \in J^-$ be an arbitrary index. (Variable x_k will be referred as the driving variable.)
If $J^- = \emptyset$, then an optimal solution is found, stop.

Step 2 *Leaving variable selection*
Let $r = \text{argmin}\{\frac{x_i}{t_{ik}} \ : \ t_{ik} > 0\}$. ($x_r$ is the leaving variable.)
If there is no r, then the problem is unbounded, stop.

Step 3 *Entering variable selection*
Let $s = \text{argmin}\{-\frac{z_j}{t_{rj}} \ : \ t_{rj} < 0, \ j \in J^+\}$. If there is no s, then let $s = k$.
Let $\theta_1 = -\frac{z_s}{t_{rs}}$ and $\theta_2 = -\frac{z_k}{t_{rk}}$.

Step 3a If $\theta_1 < \theta_2$, then make a pivot on (r, s), x_r enters the basis. Let $J^+ = \{j : z_j \geq 0\}$. The driving variable is not changed. Go to **Step 2**.

Step 3b If $\theta_1 \geq \theta_2$, then make a pivot on (r, k), x_k enters the basis. Let $J^+ = \{j : z_j \geq 0\}$. Go to **Step 1**.

Note that in Step 3b, the set of dual feasible indices J^+ is strictly increasing and it does not decrease (but may increase) in Step 3a. That is why this algorithm is called a *monotonic build-up* simplex method.

The pivot selection rule can be illustrated by Figure 2. In this scheme pivots are made in position $(-)$ in Step 3a and in position $[+]$ in Step 3b of MBU.

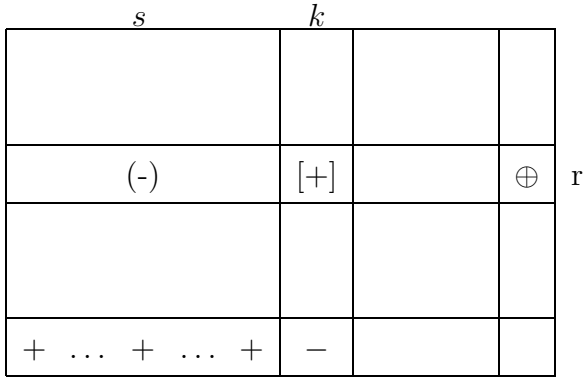

Figure 2.

Note that during some pivot sequence while the driving variable is fixed, the basis may become infeasible. Interestingly as was shown in [1], the ratio test in Step 2 is automatically restricted to the feasible part of the basis. In Step 3b, when the driving variable enters the basis, the primal feasibility of the whole basis is restored. Therefore one always has a primal feasible basis in Step 1. The validity of these statements and the correctness of the algorithm are proved in [1].

We further note that the algorithm has a strict monotone build-up feature. Obviously, J^+ is monotonically extending in Step 3b and at the same time the primal feasibility is restored. Interestingly, J^+ can be dynamically extended even if the driving variable is kept unchanged, and the dual feasibility of these variables is preserved in the subsequential pivots in Step 3a.

This build-up structure resembles slightly to recursive rules, especially to Bland's [8, 9] recursion and to the Edmonds–Fukuda [24] rule. But it can be easily seen that the two pivot rules are different. The relation between MBU, EPSA and also the shadow vertex

algorithm (the self–dual parametric simplex method and to Lemke's method) is more direct. Here these algorithms will be presented as a special implementation of MBU to a properly modified problem. This claim becomes clearer if we look at the dual version of MBU.

We have seen that MBU goes through basic solutions that are in general neither primal nor dual feasible, but some of these solutions are primal feasible. The algorithm solves the original problem by subsequently solving its subproblems. Having a primal feasible solution in Step 1 we identify the actual subproblem that is solved (collect the dual feasible variables). Then by choosing the driving variable, the size of the subproblem to be solved is increased. This subproblem is solved by relaxing primal feasibility, but during the procedure, the size of the subproblem might be increased in Step 3a if new variables became dual feasible. So the size of the subproblem dynamically grows. MBU is based on the repeated application of this idea. This demonstrates MBU's monotonic build up feature.

It is still not clear why it is a *simplex* algorithm since the feasibility may be lost here, whereas it is preserved in simplex algorithms. Actually it can be shown that MBU is equivalent to a specific primal simplex variant. Namely, a *primal feasible basic solution* can be associated to every basic solution that occurs in the procedure. More precisely, if we would make a pivot on position (r, k) in Step 3, then the resulting basic tableau (basis) would be primal feasible.

Having a dual feasible basis, the pivot rule of the dual MBU is as follows.

Dual MBU - The Pivot Rule

Step 0 Let $J^+ = \{i : x_i \geq 0\}$, $J^- = J \setminus J^+ = \{i : x_i < 0\}$.

Step 1 Let $k \in J^-$ be an arbitrary index. (Variable x_k will be referred as the driving variable.)
If $J^- = \emptyset$, then an optimal solution is found, stop.

Step 2 *Entering variable selection*
Let $s = \text{argmin}\{-\frac{z_j}{t_{kj}} \ : \ t_{kj} < 0\}$. ($x_s$ is the entering variable.)
If there is no s, then the problem is unbounded, stop.

Step 3 *Leaving variable selection*
Let $r = \text{argmin}\{\frac{x_i}{t_{is}} \ : \ t_{is} > 0, \ i \in J^+\}$. If there is no r, then let $r = k$.
Let $\theta_1 = \frac{x_r}{t_{rs}}$ and $\theta_2 = \frac{x_k}{t_{ks}}$.

Step 3a If $\theta_1 < \theta_2$, then make a pivot on (r, s), x_r leaves the basis. Let $J^+ = \{i : x_i \geq 0\}$. The driving variable is not changed. Go to **Step 2**.

Step 3b If $\theta_1 \geq \theta_2$, then make a pivot on (k, s), x_k leaves the basis. Let $J^+ = \{i : x_i \geq 0\}$. Go to **Step 1**.

Another interesting feature of MBU (as a generalization of the *shadow vertex idea* – optimization on a plane) is that it can be interpreted geometrically as an *active set method*. Let us consider the dual variant of MBU. A subproblem (while a driving variable is fixed) can be interpreted as to maximize the dual objective on the intersection of the hyperplane defined by the driving variable and the optimal cone. After that, a new driving variable (and so a new hyperplane) is selected, and the procedure repeats.

The shadow vertex algorithm is mostly known by its geometrical interpretation. We present here an algebraic form, which was also discussed in Borgwardt [11].

Shadow Vertex Algorithm

Initialization: Let a dual feasible basis $\tilde{B}$ be given. Let $0 < \tilde{v} \in R^m$ be an arbitrary positive vector, and denote $v = A_{\tilde{B}}\tilde{v}$. Let us append this new vector to the current dual feasible basic tableau. (In the sequel vector v will play the role of the solution column, and the variable associated with vector $(-b)$ will play the role of the driving variable.)

Shadow Vertex Algorithm - The Pivot Rule

Step 1 The variable associated with vector $(-b)$ is the driving variable. It is the only dual infeasible variable and the solution with respect to v is primal feasible.

Step 2 Apply MBU in this situation.

As it is known, primal feasibility (optimality) holds at each step for a linear combination of the two right hand sides b and v. This property is clear from the simplex interpretation of MBU as well.

Exterior Point Simplex Algorithm (EPSA)

EPSA was proposed by Paparrizos [68, 69]. By specializing this method Paparrizos obtained nice results for the assignment problem [70]. EPSA can be presented as follows.

Initialization: Let a dual feasible basis $\tilde{B}$ be given. Let $0 < \tilde{v} \in R^m$ be an arbitrary positive vector, and denote $v = A_{\tilde{B}}\tilde{v}$. Let us append this new vector $\tilde{v}$ to the current dual feasible basic tableau as the $(n+1)$-th column (for the original problem we have $a_{n+1} = v$). Let the corresponding reduced cost value be zero (for the original problem we have $z_{n+1} = c_{\tilde{B}}^T A_{\tilde{B}}^{-1} v = c_{\tilde{B}}^T \tilde{v}$).

EPSA - The Pivot Rule

Step 1 Make a pivot on $(r, n+1)$, where $r = \text{argmin}\{\frac{x_i}{t_{i,n+1}}\}$. (The new basis is dual feasible, and x_{n+1} is the only primal infeasible variable.)
The driving variable is $v = a_{n+1}$.

Step 2 Apply the dual MBU in this situation.

It is obvious that, after Step 1 we have an almost optimal basis except one primal infeasible variable.

EPSA and the shadow vertex algorithm have the common idea to introduce an auxiliary vector to force primal feasibility. The main difference is that an additional pivot is performed in EPSA. (A pivot in the $v-$row of the tableau.) Based on this observation it is easy to see that EPSA and the shadow vertex algorithm are also closely related.

EPSA, MBU and Finite Combinatorial Rules

We discussed some monotonic build-up schemes for LP in the second section. In fact any recursive type algorithm [41, 79, 24] can be interpreted as an algorithm that solves subproblems with monotonically increasing size. The simplex variants like Bland's minimal index rule and Edmonds–Fukuda rule preserve feasibility while MBU does not. However a feasible basis can always be associated with the infeasible basis. In contrast the criss–cross methods [94, 79, 80, 87] are real exterior "simplex–like" methods and they do not need any feasible basis to start with. Primal and dual feasibilities are guaranteed only at the last pivot step in which the optimal solution is found. Although these methods visit vertices which are neither primal nor dual feasible, they are essentially different from MBU. One of the main differences is that there is no simplex interpretation for the criss–cross methods.

Via the shadow vertex algorithm and the EPSA, it is clear that MBU has a strong background in parametric programming. In fact it can be interpreted as a repeated reparametrization of the right hand side with the actual driving variable (vector).

The last problem that we want to address corresponding parametric rules is the finiteness property and exponential examples. For an exponential example for the shadow vertex algorithm see Goldfarb [34]. Paparrizos [69] presents an exponential example for EPSA. Since both are special cases of MBU, it follows that MBU is also exponential in the worst case. However, it is easy to check that degeneracy handling methods are applicable to make these algorithms finite. While the driving variable does not change, one may use the dual lexicographic method in the inner cycle to make MBU (EPSA) finite. Of course the minimal index rule can also be applied, but to prove the finiteness in this case needs some extra effort.

Magnanti and Orlin's Parametric Algorithm

Magnanti and Orlin [56] presented a parametric programming algorithm and specialized the algorithm to derive anti–cycling pivot rules for LP. This algorithm is again closely related to the other parametric rules in this section. Let $d \in R^n$ be a parameter vector. The following parametric LP is considered.

$$(P(\theta)) \qquad \min\{(c^T + \theta d^T)x : Ax = b,\ x \geq 0\}.$$

Now we consider the modified problem, where an extra row (d) and an extra variable (x_d) is introduced:

$$(P(\theta)) \qquad \min\{c^T x + \theta x_d : d^T x + x_d = 0,\ \ Ax = b,\ x \geq 0\}.$$

Note that if we have a basis B with basis variables x_B for (P), then obviously (x_B, x_d) provides a basis for $(P(\theta))$, and the simplex tableau is extended with the extra row (d) and a column (unit vector).

Magnanti and Orlin's Pivot Rule

Step 0 Let B be an optimal basis for $(P(\theta))$ with a given θ. Let the initial basis defined by (x_B, x_d). Keep the unrestricted variable x_d as the driving variable.

Step 1 Apply the entering and leaving variable selection rule of dual MBU in this situation.

Magnanti and Orlin shows that using this pivot rule $(P(\theta))$ can be solved for each θ. Their anti–cycling pivot rules are special implementations of the above parametric rule. One might perturb either the objective vector c, or the parameter vector d or both. This way they presented the three different anti–cycling lexicographic rule.

Transition Node Pivoting (TNP)

To conclude this section we mention the so-called Transition Node Pivot (TNP) rule suggested by Geue [32] and studied further by Gal and Geue [30]. The TNP rule applies only when degeneracy occurs. The rule is activated at the step before degeneracy occurs.

Assume we are facing the situation that the current basis is nondegenerate with an entering variable k, and after this pivot the next basis becomes primal degenerate. The set of indices for which the minimum ratio is attained will be denoted by $J_{\min}$.

TNP Pivot Rule

Step 0 The entering variable k is given, where the next basis is degenerate. Let t be another nonbasic variable index (t is referred as a *transition column*).
Let $r = \operatorname{argmax}\{\frac{t_{it}}{t_{ik}} \ : \ i \in J_{\min}\}$ (x_r is the leaving variable).

Step 1 Make a pivot on (r, k).

If the basis is nondegenerate continue with any simplex pivot rule.

Step 2 *Entering variable selection*
Choose any dual infeasible variable k to enter the basis.

If there is no entering variable, then the problem is solved, stop.

Step 3 *Leaving variable selection*
Make the ratio test.

If $|J_{\min}| = 1$ then denote the element in $J_{\min}$ by r. Go to **Step 1**.

Step 3a Let $r = \operatorname{argmax}\{\frac{t_{it}}{t_{ik}} \ : \ i \in J_{\min}\}$ (x_r is the leaving variable). Go to **Step 1**.

Similarly as with the previous parametric rules the TNP rule can be combined with the minimal index or lexicographic rule to guarantee finiteness of the procedure. A "perturbation" column as the transition column can be introduced as with the lexicographic rule. Then the reduced cost of this extra column strictly increases, and hence cycling becomes impossible.

4 Pivot Rules Related to Interior Point Methods

In this section we will discuss some recent pivot rules for the simplex method which are in one way or another related to interior point methods. Since Karmarkar [43] first proposed an interior method which solves linear programming problems in polynomial time and this method is reported to perform superior to the simplex method for large scale problems, there have been a large number of research papers devoted to that topic. Interior point methods allow the iterative points to go inside the polytope and they are believed to be able to use more global information and therefore to avoid "myopia" of the simplex method. Although it is still unclear that we may succeed in finding a polynomial simplex algorithm by using interior point type methodology, we think it is definitely a new field deserving further investigation. In this section mainly three pivot rules will be discussed. They are: Todd's rule [84] which was obtained as a limit of a projective interior point method, the minimal volume simplex rule of Roos [71] and the dual interior primal simplex method of Tamura, Takehara, Fukuda, Fujishige and Kojima [78].

Todd's Dantzig–Wolfe Like Pivot Rule

The pivot rule proposed by Todd [84] is closely related to the pivot rules presented by Klotz [51]. These rules are based on the reduced cost scaling as it was earlier used by Harris [37] in the DEVEX code and also studied by Kalan [42]. Reduced cost scaling with various scaling methods and their effect on the numerical performance of the simplex algorithm is extensively studied by Klotz [51]. Todd's pivot rule is based on a variant of Karmarkar's interior point method (cf. Todd [84]). The method is described as follows.

We consider the linear programming in the following form

$$\begin{aligned} \min \quad & c^T x & & \\ \text{s.t.} \quad & Ax & = & \ 0 \\ & g^T x & = & \ 1 \\ & x & \geq & \ 0, \end{aligned}$$

where g is a certain vector.
Let x^k be an interior point in the feasible region of the above problem. Let $X = X_k := diag(x^k)$ denote the diagonal matrix with diagonal entries the coordinates of x^k. Define

the oblique projection matrix

$$P = P_k := I - X^2 A^T (A X^2 A^T)^{-1} A.$$

It is easy to see that $AP = 0$ and $P^2 = P$.

Let the oblique projection of c and g be given as

$$\tilde{c} := P^T c \text{ and } \tilde{g} := P^T g.$$

The relaxed problem is defined by

$$\begin{array}{lrcl} \text{(RP)} & \min & \tilde{c}^T x & \\ & \text{s.t.} & \tilde{g}^T x & = & 1 \\ & & x & \geq & 0. \end{array}$$

Note that (RP) is a continuous knapsack problem and it can be solved easily. If (RP) has an optimal solution w, let $\tilde{w} := Pw$. If (RP) is unbounded with an unbounded ray direction t, let $\tilde{t} := Pt$. In the first case, let $x^{k+1} := (1 - \lambda_k)x^k + \lambda_k \tilde{w}$, $\lambda_k \in (0, 1]$. In the second case, let $x^{k+1} := x^k + \mu_k \tilde{t}$, $\mu_k > 0$. With proper choice of λ_k or μ_k, and using Karmarkar's potential function, Todd [84] proved that this method has the same convergence property as Karmarkar's method.

Now, if x^k is a nondegenerate basic feasible solution with basis B, a similar procedure can be applied. This results in the following pivot rule.

Todd's Projective Pivot Rule

Step 1 Let B be a primal feasible basis and suppose P solves $A_B^T P = c_B$ and σ solves $A_B^T \sigma = X_B^{-1} e$ where X_B is the diagonal matrix with diagonal entries the values of the basic variables and e is the vector with each component equal to 1. For each $j \notin B$, compute the reduced cost $z_j := c_j - P^T a_j$ and $\tilde{g}_j := \sigma^T a_j$.

Step 2 If there is an r with $z_r < 0$ and $\tilde{g}_r = 0$, select it as the entering basis variable index.

Otherwise, select $r = arg\max\{\frac{z_j}{\tilde{g}_j} : z_j < 0, \tilde{g}_j < 0\}$.

Else, select $r = \text{argmin}\{\frac{z_j}{\tilde{g}_j} : z_j < 0, \tilde{g}_j > 0\}$.

If none of these applies, the current basis is optimal, stop.

Step 3 Select the leaving variable according to the usual minimum ratio test. If no leaving variable is found, the problem is unbounded, otherwise go to **Step 1**.

Reduced cost scaling is used in Todd's pivot rule. More analyses can be found in Todd's paper [84]. We note here that in case of degeneracy additional rules need to be adapted to prevent cycling.

Minimal Volume Simplex Pivot Rule

The *minimal volume simplex* pivot rule is suggested in an unpublished paper of Roos [71]. In Roos' paper, the primal problem (P) is considered. As in Karmarkar's approach, he assumes that the optimal value is known to be zero (this assumption is standard in the analysis of projective methods). Moreover, all basic solutions are assumed to be nondegenerate. For a given basis B, let A_B be the submatrix of A and c_B be the subvector of c corresponding to the index set (basis) B. Now, consider a cone $\mathcal{C}_\mathcal{B}$ in R^m associated with the basis B as follows:

$$\mathcal{C}_\mathcal{B} := \{y \in R^m : \ A_B^T y \leq c_B\}.$$

If the rows of A_B^{-1} are given by $y_1, y_2, ..., y_m$, then it can be shown that

$$\mathcal{C}_\mathcal{B} = \{y_B - \sum_{i=1}^{m} \lambda_i y_i : \ \lambda_i \geq 0, i = 1, 2, ..., m\},$$

where $y_B^T := c_B^T A_B^{-1}$.

Moreover, the vectors y_i, $i = 1, 2, ..., m$, are the rays of the cone $\mathcal{C}_\mathcal{B}$. By the definition of the tableau, it is obvious that $t_{ij} = y_i^T a_j$, $i = 1, 2, ..., m$, $j = 1, 2, ..., n$.

The problem $\max\{y^T b : \ y \in \mathcal{C}_\mathcal{B}\}$ is bounded from above if and only if $y_i^T b \geq 0$ for $i = 1, 2, ..., m$, and this corresponds to the feasibility of the basis B. Suppose the feasible basis B is not optimal. Remember that the optimal value was assumed to be zero, this means that $y_B^T b > 0$. Therefore the intersection of the cone $\mathcal{C}_\mathcal{B}$ and the half space $\{y : \ y^T b \geq 0\}$ forms a nontrivial simplex. We denote this simplex by S_B. It is shown in Roos [71] that the volume of S_B can be computed as

$$vol(S_B) = \frac{1}{n!} \frac{(c^T x)^n}{\prod x} \frac{1}{\mid \det A_B \mid},$$

where Πx denotes the product of positive entries of x. The minimal volume pivot rule of Roos seeks for the next feasible basis with the minimal volume of such simplex. Note here that the simplex corresponding to the optimal basis has zero volume.

To calculate the volume of this simplex does not need much effort. If the pivot element is denoted by t_{rs}, then it follows that $\mid \det A_{B'} \mid = t_{rs} \mid \det A_B \mid$ with B' the next feasible basis. Hence Roos' pivot rule looks for the entering variable index r with $z_r < 0$ and the feasible pivot element t_{rs} such that

$$\frac{1}{t_{rs}} \frac{(c^T x')^n}{\prod x'}$$

is minimal, where x' denotes the basic feasible solution corresponding to the next feasible basis B'.

Observe that this formula strongly resembles Karmarkar's potential function. However, it is still unclear at this stage how to cope with the degenerate case. Also, more properties of this method need to be investigated. In formal presentation, Roos' pivot rule is as follows.

Roos' Pivot Rule

Step 1 For an LP problem in the primal form having optimal value zero, let B be a feasible basis. Let $y^T := c_B^T A_B$ and the reduced cost vector $z^T := c^T - y^T A$.

Step 2 For all possible pivot on position (r, s) with $z_r < 0$ preserving primal feasibility, compute

$$\frac{1}{t_{rs}} \frac{(c^T x')^n}{\prod x'}$$

where x' is the next basic feasible solution, and select the pair (r, s) with the minimal value. Pivot on that position and go to **Step 1**.

Concerning Roos' unpublished paper [71] we note that he also presented another pivot rule, the so-called *maximum distance* pivot rule. Here the variable is selected to enter the basis for which the corresponding hyperplane (in the dual formulation) is the farthest from the present vertex.

Dual Interior Primal Simplex Method (DIPS)

Finally in this section we will introduce the socalled Dual Interior Primal Simplex (DIPS) method developed by Tamura, Takehara, Fukuda, Fujishige and Kojima [78]. This method is also referred sometimes as *Stationary Ball Method*. DIPS is a modification of the gravitational method of Chang and Murty [64] and Murty [62]. The gravitational method can be explained as follows.

We consider the dual problem $(D) : \max\{b^T y : A^T y \leq c\}$. Suppose the interior of $\{y : A^T y \leq c\}$ is nonempty, then we can place a ball with certain radius r inside that region. Now, suppose the objective direction b is vertical, the ball will then fall down by the gravitational force until it touches some boundary of that region and rolls down while minimizing its potential energy. If we then reduce the radius of the ball subsequentially to zero, it will fall and roll down until it reaches the "bottom" of the feasible region (optimal point).

If an initial ball in a stationary position is known, and the radius of the ball is reduced discretely to zero, it can be shown under the nondegeneracy assumption, that the trajectory of the centers of the ball is a piece-wise linear locus. The gravitational method of Chang and Murty is to follow this central trajectory until certain precision requirements are satisfied. Their method is finite.

The DIPS method is based on a similar idea. An essential observation (Chang and Murty [15], Murty [64], and Tamura *et al.* [78]) is that a subset of constraints touched by a ball in its stationary position (after falling down) forms a feasible basis for the primal problem (P). Moreover, two adjacent break points in the central trajectory result in two different but adjacent bases of the primal problem. For detailed proofs, one refers to Tamura *et al.* [78]. This trajectory is related to the "inverse barrier path" and is discussed by Den Hertog, Roos and Terlaky [40].

To derive a pivot rule some additional assumptions were needed. In the original paper it was assumed that $||a_j|| = 1$ for all j and the LP problem is dual nondegenerate. Note that the first assumption is not restrictive since it can be easily obtained by scaling; moreover, the resulted pivot rule can be presented without the second assumption as will be seen later.

Let y^i, $i = 1, 2, ...$, denote the break points of the piecewise linear central trajectory and B_i the feasible basis corresponding to the dual constraints reached by the ball with center y^i and with the maximum radius r_i of that position. Furthermore, if x^i denotes the basic feasible solution corresponding to the basis B_i, $i = 1, 2, ...$, it is shown by Tamura *et al.* that

$$c^T x^{i+1} \leq c^T x^i,\ i = 1, 2, ...$$
$$b^T y^{i+1} \geq b^T y^i,\ i = 1, 2, ...$$

and

$$\lim_{i\to\infty} c^T x^i = \lim_{i\to\infty} b^T y^i.$$

This explains the name of the method (Dual Interior Primal Simplex).

More specifically, the rule for selecting the entering basis variable in the primal simplex version can be stated explicitly as follows[2].

Dual Interior Pivot Rule

Step 1 For a primal feasible tableau $T(B)$, where B corresponds to indices of dual constraints touched by the ball with center y and radius r in the dual space, if there is a nonbasic index k such that $a_k^T y - c_k = -\|a_k\| r$ (i.e. the hyperplane corresponding to index k also touches the ball) then we set the entering index s to be the minimal index among such nonbasic variables; otherwise select

$$s = \text{argmin}_{c_k - a_k^T z < 0} \{ \frac{(a_k^T y - c_k + \|a_k\| r) + (c_k - a_k^T w)}{a_k^T y - c_k + \|a_k\| r} \}$$

where $w^T := c_B^T A_B^{-1}$, a_k denotes the kth column of A, c_k the kth coordinate of c (therefore $c_k - a_k^T w$ is the reduced cost of x_k).

Step 2 Select the leaving basis variable according to the usual simplex ratio test. Update the basis B and go to **Step 1**.

[2]This formulation is given to the authors by Professor Tamura [77].

The DIPS method is reported to have similar numerical behavior as the maximum decrease pivot rule, whereas the complexity to perform each pivot is much less. Also, relations between the DIPS method and the parametric programming technique were discussed in [78]. This concludes the DIPS method and this section.

There are many interesting results related to pivot rules, as well as many open questions. In the next section, we will conclude the paper with some remarks.

5 Miscellaneous Remarks

Although interior point methods have been most intensively studied in the last years, several pivot rules were developed for linear programming. We have surveyed three classes of pivot rules – combinatorial, parametric type and those derived from interior point methods.

An interesting (and efficient) application of the minimal index rule and the criss–cross method is presented by Avis and Fukuda [4]. By reversing these rules an algorithm is derived that enumerates all the feasible bases (vertices of a polyhedra) and all the bases without extra storage requirement.

Concerning combinatorial pivot rules note that there are a great number of pivot rules developed for special classes of network optimization. Most of them were never studied or generalized for general linear programming. To discuss these rules is beyond the scope of the present paper. We also note that it is interesting to find a simple finiteness proof for Todd's simplex pivot rule for oriented matroid programming, or to find some other pivot rules belonging to this category.

Interestingly, only simplex methods based on parametric programming – the shadow vertex algorithm and Lemke's method – are proved to be polynomial in the expected number of pivot steps [11, 85]. It is still unknown if the other rules have this property as well. To the best knowledge of the authors only one attempt was made to examine the average number of steps of combinatorial pivot rules [26].

In the extensive literature of interior point methods one can find some papers discussing the relation between interior point methods and the simplex methods. Among them, there are papers by Chiu and Ye [16] and by Stone and Tovey [76]. In the last paper the simplex method and interior point methods were presented as iteratively reweighted least squares methods.

Sehti and Thompson [75] proposed the Pivot and Probe method where the simplex method was combined with a build–up structure. This algorithm appeared before the theory of interior point methods became the central area for research in linear programming.

Another recent trend to improve the efficiency of LP algorithms is to combine interior point algorithms with the simplex method. In this approach some interior point method is first applied to generate a sufficiently good interior solution, then the algorithm switches

to the simplex method to find an optimal basis. This type of methods are discussed in [52, 57, 58, 7, 90].

Although to discuss "practical pivot rules" and implementation strategies is beyond the scope of this paper we mention here some recent papers discussing practical problems and state of the art implementations. The paper of Forrest and Tomlin [23] discusses the implementation of the simplex method in IBM's Optimization Subroutine Library. The papers of Bixby [5, 6] discuss the implementational issues of the CPLEX package. Practical degeneracy handling strategies are presented and tested in [12, 17, 35, 33, 38, 73]. Most of the efficient packages [6, 23] utilize the advantages of DEVEX rules (which are based on reduced cost scaling) [37, 42, 51] and a composite objective function [6, 23, 21] (which is a linear combination of the original objective function and a linear function measuring infeasibility).

The steepest edge simplex algorithm also shows practical efficiency as it is presented in [35]. Boyd [11] presents some modifications of the steepest edge algorithm for resolving degeneracy in combinatorial linear programming problems. He proves that the steepest ascent algorithm is finite. However, the solution of a quadratic programming problem is needed to find the steepest ascent direction.

Towards degeneracy in linear programming and some other related problems, there is a relatively new approach called *theory of degeneracy graphs* studied by Gal *et al.* [29]. In that theory, a degenerate vertex is associated with a graph. Each node of the graph represents a basis corresponding to the vertex. An edge between two nodes on the graph implies that a pivot step can be performed to move from one basis to the other. To study this graph will certainly give insight about the structure of the degenerate vertex.

To conclude the paper we note that the hardest and long standing open problems in the theory of linear programming are still concerned with pivot methods. Those include the d–step conjecture [50] and the question whether there exists a polynomial pivot rule or not. For the last problem Zadeh's rule [91] might be a candidate. At least it is still not proved to be exponential in the worst case.

Acknowledgements

The authors are grateful to the anonymous referees and Hans Frenk for the constructive comments on the first version of the paper.

References

[1] K.M. Anstreicher and T. Terlaky, A Monotonic Build Up Simplex Algorithm, Report 91–82, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1990).

[2] I. Adler and N. Megiddo, A Simplex Algorithm Whose Average Number of Steps is Bounded Between Two Quadratic Functions of the Smaller Dimension, Journal of the Association of Computing Machinery 32(1985)891–895.

[3] D. Avis and V. Chvátal, Notes on Bland's Rule, Mathematical Programming Study 8(1978)24–34.

[4] D. Avis and K. Fukuda, A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra, Research Report B-237, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan (1991).

[5] R.E. Bixby, Implementing the Simplex Method: The Initial Basis, Report TR90–32, Department of Mathematical Sciences, Rice University, Texas, USA (1991).

[6] R.E. Bixby, The Simplex Method: It Keeps Getting Better, Lecture at the 14th International Symposium on Mathematical Programming, Amsterdam, The Netherlands (1991).

[7] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten and D.F. Shanno, Very Large–Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods, Report, Department of Mathematical Sciences, Rice University, Texas, USA (1991).

[8] R.G. Bland, New Finite Pivoting Rules for the Simplex Method, Mathematics of Operations Research 2(1977)103–107.

[9] R.G. Bland, A Combinatorial Abstraction of Linear Programming, Journal of Combinatorial Theory (Ser. B) 23(1977)33–57.

[10] K.H. Borgwardt, *The Simplex Method: A Probabilistic Analysis,* Algorithms and Combinatorics, Vol. 1. (Springer Verlag, 1987).

[11] E.A. Boyd, Resolving Degeneracy in Combinatorial Linear Programs: Steepest Edge, Steepest Ascent and Closest Ascent, Report, Department of Mathematical Sciences, Rice University, Texas, USA (1992).

[12] N. Cameron, Stationarity in the Simplex Method, Journal of the Australian Mathematical Society 43(1987)137–142.

[13] Y.Y. Chang, Least Index Resolution of Degeneracy in Linear Complementarity Problems, Technical Report 79–14, Department of Operations Research, Stanford University, Stanford, California, USA (1979).

[14] Y.Y. Chang and R.W. Cottle, Least Index Resolution of Degeneracy in Quadratic Programming, Mathematical Programming 18(1980)127–137.

[15] S.Y. Chang and K.G. Murty, The Steepest Descent Gravitational Method for Linear Programming, Discrete and Applied Mathematics, 25(1989)211–239.

[16] S.S. Chiu and Y. Ye, Simplex Method and Karmarkar's Algorithm: A Unifying Structure, Technical Report, Engineering Economic Systems Department, Stanford University, Stanford, California, USA (1985).

[17] M. Cirina, Remarks on a Recent Simplex Pivoting Rule, Methods of Operations Research 49(1985)187–199.

[18] J. Clausen, A Note on Edmonds–Fukuda Pivoting Rule for the Simplex Method, European Journal of Operations Research 29(1987)378–383.

[19] R.W. Cottle, The Principal Pivoting Method Revisited, Technical Report SOL 89–3, Stanford University, Stanford, California, USA (1989).

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

[21] H.A. Eiselt and C.-L. Sandblom, External Pivoting in the Simplex Algorithm, Statistica Neerlandica 39(1985)327–341.

[22] J. Folkman and J. Lawrence, Oriented Matroids, Journal of Combinatorial Theory (Ser. B) 25(1978)199–236.

[23] J.J.H. Forrest and J.A. Tomlin, Implementing the Simplex Method for the Optimization Subrutin Library, IBM Systems Journal 31(1992)11–25.

[24] K. Fukuda, Oriented Matroid Programming, Ph.D. Thesis, Waterloo University, Waterloo, Ontario, Canada (1982).

[25] K. Fukuda and T. Matsui, On the Finiteness of the Criss–Cross Method, European Journal of Operations Research 52(1991)119–124.

[26] K. Fukuda and M. Namiki, Two Extremal Behavior of the Criss–Cross Method for Linear Complementarity Problems, Research Report B-241, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan (1991).

[27] K. Fukuda and T. Terlaky, A General Algorithmic Framework for Quadratic Programming and a Generalization of Edmonds–Fukuda Rule as a Finite Version of Van de Panne–Whinston Method, Mathematical Programming (1989) (to appear).

[28] K. Fukuda and T. Terlaky, Linear Complementarity and Oriented Matroids, Journal of the Operational Research Society of Japan (1990) (to appear).

[29] T. Gal, Degeneracy Graphs – Theory and Application: A State-of-the-Art Survey, Report No. 142, Fern Universität Hagen, Germany (1989).

[30] T. Gal and F. Geue, The use of the TNP–Rule to Solve Various Degeneracy Problems, Report No. 149a, Fern Universität Hagen, Germany (1990).

[31] S. Gass and Th. Saaty, The Computational Algorithm for the Parametric Objective Function, Naval Research Logistics Quarterly 2(1955)39–45.

[32] F. Geue, Eine neue Pivotauswahlregel und die durch sie induzierten Teilgraphen des positiven Entartungsgraphen, Report No. 141, Fern Universität Hagen, Germany (1989).

[33] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, A Practical Anti–Cycling Procedure for Linearly Constrained Optimization, Mathematical Programming 45(1989)437–474.

[34] D. Goldfarb, Worst Case Complexity of the Shadow Vertex Simplex Algorithm, Report, Department of Industrial Engineering and Operations Research, Columbia University, USA (1983).

[35] D. Goldfarb and J.J.H. Forrest, Steepest Edge Simplex Algorithm for Linear Programming, IBM Reserch Report, June 1991, T.J. Watson Research Center, Yorktown Heights, USA (1991).

[36] M. Haimovitz, The Simplex Method is Very Good! – on the Expected Number of Pivot Steps and Related Properties of Random Linear Programs, Report, Columbia University, New York, USA (1983).

[37] P.M.J. Harris, Pivot Selection Methods for the Devex LP Code, Mathematical Programming 5(1973)1–28.

[38] B. Hattersly and J. Wilson, A Dual Approach to Primal Degeneracy, Mathematical Programming 42(1988)135–176.

[39] D. Den Hertog, C. Roos and T. Terlaky, The Linear Complementarity Problem, Sufficient Matrices and the Criss–Cross Method, Report, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1990).

[40] D. Den Hertog, C. Roos and T. Terlaky, The Inverse Barrier Method for Linear Programming, Report No. 91–27, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1991).

[41] D. Jensen, Coloring and Duality: Combinatorial Augmentation Methods, Ph.D. Thesis, School of OR and IE, Cornell University, Ithaca, N.Y., USA (1985).

[42] J.E. Kalan, Machine Inspired Enhancements of the Simplex Algorithm, Technical Report CS75001-R, Computer Science Department, Virginia Polytechnical University, Blacksburg, Virginia, USA (1976).

[43] N. Karmarkar, A New Polynomial–Time Algorithm for Linear Programming, Combinatorica 4(1984)373–395.

[44] L.G. Khachian, Polynomial Algorithms in Linear Programming, Zhurnal Vichislitelnoj Matematiki i Matematischeskoi Fiziki 20(1980)51–68 (in Russian), USSR Computational Mathematics and Mathematical Physics 20(1980)53–72 (English translation).

[45] E. Klafszky and T. Terlaky, Remarks on the Feasibility Problems of Oriented Matroids, Annales Universitatis Scientiarum Budapestiensis de Rolando Eötvös Nominatae, Sectio Computatorica, Tom. VII (1987)155–157.

[46] E. Klafszky and T. Terlaky, Variants of the Hungarian Method for Solving Linear programming problems, Math. Oper. und Stat. ser. Optimization 20(1989)79–91.

[47] E. Klafszky and T. Terlaky, Some Generalizations of the Criss–Cross Method for Quadratic Programming, Math. Oper. und Stat. ser. Optimization (1989) (to appear).

[48] E. Klafszky and T. Terlaky, Some Generalizations of the Criss–Cross Method for the Linear Complementarity Problem of Oriented Matroids, Combinatorica 9(1989)189–198.

[49] V. Klee and G.J. Minty, How Good is the Simplex Algorithm? in: *Inequalities–III,* ed. O. Shisha (Academic Press, New York, 1972) p. 159–175.

[50] V. Klee and P. Kleinschmidt, The d–step Conjecture and Its Relatives, Mathematics of Operations Research 12(1987)718–755.

[51] E. Klotz, Dynamic Pricing Criteria in Linear Programming, Technical Report SOL. No. 88–15, Systems Optimization Laboratory, Stanford University, Stanford, California USA (1988).

[52] K.O. Kortanek and J. Zhu, New Purification Algorithms for Linear Programming, Naval Research Logistics Quarterly 35(1988)571–583.

[53] H.P. Künzi and H. Tzschach, The Duoplex–Algorithm, Numerische Mathematik 7(1965)222–225.

[54] C.E. Lemke, Bimatrix Equilibrium Points and Mathematical Programming, Management Science 11(1965)681–689.

[55] I. Lustig, The Equivalence of Dantzig's Self–Dual Parametric Algorithm for Linear Programs to Lemke's Algorithm for Linear Complementarity Problems Applied to Linear Programming, Technical Report SOL 87–4, Department of Operations Research, Stanford University, Stanford, California, USA (1987).

[56] T.L. Magnanti and J.B. Orlin, Parametric Linear Programming and Anti–Cycling Pivoting Rules, Mathematical Programming 41(1988)317–325.

[57] N. Megiddo, On Finding Primal– and Dual–Optimal Bases, RJ 6328 (61997), IBM Almaden Research Center, San Jose, California, USA (1988).

[58] N. Megiddo, Switching from a Primal–Dual Newton Algorithm to a Primal–Dual (Interior) Simplex–Algorithm, RJ 6327 (61996), IBM Almaden Research Center, San Jose, California, USA (1988).

[59] K.G. Murty, A Note on a Bard Type Scheme for Solving the Complementarity Problem, Opsearch 11 (2–3)(1974)123–130.

[60] K.G. Murty, *Linear and Combinatorial Programming,* (Krieger Publishing Company, Malabar, Florida, USA, 1976).

[61] K.G. Murty, Computational Complexity of Parametric Linear Programming, Mathematical Programming 19(1980)213–219.

[62] K.G. Murty, The Gravitational Method of Linear Programming, Technical Report No. 86–19, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109–2117, USA (1986).

[63] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming* (Sigma Series in Applied Mathematics, Vol. 3, Heldermann Verlag, Berlin, 1988).

[64] K.G. Murty, A New Interior Variant of the Gradient projection Method for Linear Programming, Technical Report No. 85–18, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109–2117, USA (1985).

[65] M. Namiki and T. Matsui, Some Modifications of the Criss–Cross Method, Research Report, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan (1990).

[66] P.–Q. Pan, Practical Finite Pivoting Rules for the Simplex Method, OR Spektrum 12(1988)219–225.

[67] K. Paparrizos, Pivoting Rules Directing the Simplex Method Through All Feasible Vertices of Klee–Minty Examples, Opsearch 26(2)(1989)77–95.

[68] K. Paparrizos, An Exterior Point Simplex Algorithm, Lecture at the EURO–TIMS conference, Belgrade, Jugoslavia (1989).

[69] K. Paparrizos, A Simplex Algorithm with a New Monotonicity Property, Working Paper, (1991).

[70] K. Paparrizos, An Infeasible (Exterior Point) Simplex Algorithm for Assignment Problems, Mathematical Programming 51(1991)45–54.

[71] C. Roos, A Pivoting Rule for the Simplex Method which is Related to Karmarkar's Potential Function, Manuscript, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1986).

[72] C. Roos, An Exponential Example for Terlaky's Pivoting Rule for the Criss–Cross Simplex Method, Mathematical Programming 46(1990)78–94.

[73] D.M.Ryan and M.R. Osborne, On the Solution of Highly Degenerate Linear Programmes, Mathematical Programming 41(1988)385–392.

[74] A. Schrijver, *Theory of Linear and Integer Programming* (John Wiley & Sons, 1987).

[75] A. Sehti and G. Thompson, The Pivot and Probe Algorithm for Solving a Linear Problem, Mathematical Programming 29(1984)219–233.

[76] R.E. Stone and C.A. Tovey, The Simplex and Projective Scaling Algorithms as Iteratively Reweighted Least Squares Methods, SIAM Review 33(2)(1991)220–237.

[77] A. Tamura, Private Communication (1991).

[78] A. Tamura, H. Takehara, K. Fukuda, S. Fujishige and M. Kojima, A Dual Interior Primal Simplex Method for Linear Programming, Journal of the Operations Research Society of Japan 31(3)(1988)413–430.

[79] T. Terlaky, A Finite Criss–Cross Method for Oriented Matroids, Journal of Combinatorial Theory (Ser. B) 42(3)(1987)319–327.

[80] T. Terlaky, A Convergent Criss–Cross Method, Math. Oper. und Stat. ser. Optimization 16(5)(1985)683–690.

[81] T. Terlaky, A New Algorithm for Quadratic Programming, European Journal of Operations Research 32(1987)294–301.

[82] M.J. Todd, Complementarity in Oriented Matroids, SIAM Journal on Algebraic and Discrete Mathematics 5(1984)467–485.

[83] M.J. Todd, Linear and Quadratic Programming in Oriented Matroids, Journal of Combinatorial Theory (Ser. B) 39(1985)105–133.

[84] M.J. Todd, A Dantzig–Wolfe Like Variant of Karmarkar's Interior–Point Linear Programming Algorithm, Operations Research 38(1990)1006–1018.

[85] M.J. Todd, Polynomial Expected Behavior of a Pivoting Algorithm for Linear Complementarity and Linear Programming Problems, Mathematical Programming 35(1986)173–192.

[86] A. Tucker, A Note on Convergence of the Ford–Fulkerson Flow Algorithm, Mathematics of Operations Research 2(2)(1977)143–144.

[87] Zh. Wang, A Conformal Elimination Free Algorithm for Oriented Matroid Programming, Chinese Annals of Mathematics 8(B1)(1987).

[88] Zh. Wang, A Modified Version of the Edmonds–Fukuda Algorithm for LP problems in the General Form, Asia–Pacific Journal of Operations Research (1989) (to appear).

[89] Zh. Wang, A General Deterministic Pivot Method for Oriented Matroid Programming, Chinese Annals of Mathematics (1990) (to appear).

[90] Y. Ye and J.A. Kaliski, Further Results on Build–Up Approaches for Linear Programming, Lecture at the ORSA/TIMS Meeting, Anaheim, California, USA (1991).

[91] N. Zadeh, What is the Worst Case Behavior of the Simplex Algorithm? Technical Report No. 27, Department of Operations Research, Stanford University, Stanford, California, USA (1980).

[92] S. Zhang, On Anti–Cycling Pivoting Rules for the Simplex Method, Operations Research Letters 10(1991)189–192.

[93] G.M. Ziegler, Linear Programming in Oriented Matroids, Technical Report No. 195, Institut für Mathematik, Universität Augsburg, Germany (1990).

[94] S. Zionts, The Criss–cross Method for Solving Linear Programming Problems, Management Science 15(7)(1969)426–445.