Operations Research
Report 2005-03

Anstreicher-Terlaky type monotonic simplex algorithms for linear feasibility problems

Bilen Filiz, Zsolt Csizmadia and Tibor Illés

May 2005

Eötvös Loránd University of Sciences
Department of Operations Research

ISSN 1215 - 5918

Anstreicher-Terlaky type monotonic simplex algorithms for linear feasibility problems

Bilen Filiz, Zsolt Csizmadia and Tibor Illés

Abstract

We define a variant of Anstreicher and Terlaky's (1994) monotonic build-up (MBU) simplex algorithm for linear feasibility problems. Under a nondegeneracy assumption weaker than the normal one, the complexity of the algorithm can be given by $m\Delta$, where Δ is a constant determined by the input data of the problem and m is the number of constraints. The constant Δ cannot be bounded in general by a polynomial of the bit length of the input data.

Realizing an interesting property of degeneracy led us to construct a new recursive procedure to handle degenerate problems. The essence of this procedure is as follows. If a degenerate pivot tableau is obtained, we define a smaller problem, restricting the pivot position to a smaller part of the tableau. On this smaller problem, we follow the same principles as before to choose the pivot position. The smaller problem is either solved completely or a new degenerate subproblem is identified. If the subproblem was solved then we return to the starting larger problem, where either we can make a nondegenerate pivot, or detect that the problem is infeasible. It is easy to see that the maximum depth of the recursively embedded subproblems is smaller than $2m$.

Upper bounds for the complexities of linear programming version of MBU and the first phase of the simplex algorithm can be found similarly under the nondegeneracy assumption.

Keywords: feasibility problem, Anstreicher-Terlaky type monotonic simplex algorithms, degeneracy, linear programming.

Mathematics Subject Classification 2000: **90C20**.

1 Introduction

The first model known in the literature to consider linear inequalities is the Fourier mechanical principle, which was also a starting point of the works of Gyula Farkas. The feasibility issues of linear were investigated by Gyula Farkas [5, 6] in the 19^{th} century. The first known computational method for special linear inequality systems was formulated by Fourier. This method was generalized to arbitrary linear systems of inequalities by Motzkin, and this method is known since as Fourier-Motzkin elimination [13].

The analysis of the solution techniques of linear systems of inequalities become a main issue of *linear optimization* developed in the middle of the 20^{th} century, and is still an interesting research area.

We consider the following *feasibility problem*

$$A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$. Without loss of generality, we may assume that $rank(A) = m$, for the feasibility check of the linear system, and the removal of the redundant equations can be done simultaneously by the Gauss-Jordan elimination. The most well known open question of feasibility problems is whether there exists a strongly polynomial pivot algorithm for solving (1). We give answer to an easier question: we show that a variant of the *Anstreicher-Terlaky type monotonic simplex algorithm* can be formulated for the problem (1), which has a complexity bound of $m\Delta$, where Δ is a constant determined by the input data of the problem. Although Δ is constant, it cannot be bounded by a polynomial of the bit length of the problem, and the analysis is also based on a special *nondegeneracy assumption*. Our result is new and up to our best knowledge no similar result is known in the literature. Indeed, since the development of Dantzig's simplex algorithm [3, 4] there was no pivot algorithm known, that has a complexity bound obtained from the analysis of the algorithm. Most pivot algorithms starting from a given basis make exponential number of iterations on the *Klee-Minty* cube [11], or on some other problem with very similar structure, before obtaining the optimal solution. For the majority of the pivot algorithms, there are no theoretical results concerning the efficiency of the method other than the proof of finiteness. On the other hand, practical experiences show ([12], page 19) that the average number of iterations required to solve a linear programming problem is $O(m)$, where m is the number of constraints of the problem. The practical experience have been supported by many probability based analysis. In other words, it has been shown that the expected number of iterations of the

simplex algorithm is polynomial. We mention the works of Borgwardt [2] and Todd [16]. More information on the topic along with numerous references can be found in the paper of Fukuda and Terlaky [7].

After Karmarkar's famous paper [9] on interior point methods, new IPM algorithms and their analysis became the most popular research area in linear programming, pushing the theoretical study and elaboration of new pivot methods into the background.[1] A rare exception is the *monotonic built-up simplex algorithm* of Anstreicher and Terlaky [1], that represents a major innovation after the former simplex type algorithms. This algorithm has the following main properties: (i) it does not necessarily keep primal feasibility during the iterations, (ii) it compares two kinds of pivot positions, (iii) in case of degeneracy it uses some kind of anti-degeneracy rule like minimal index rule, (iv) if the feasibility of a primal variable is violated it is restored after a pivot is made on the so-called leading variable.

In this paper, we define a new algorithm for solving linear feasibility problems following the main idea of the monotonic built-up simplex method of Anstreicher and Terlaky [1]. We elaborated both the primal and dual versions of our algorithm and both of them have the property that once a variable becomes feasible, it remains so during the algorithm.

Our choice of pivot position differs from the usual ones from the literature for feasibility (linear programming) problems.

We show that our algorithm – under a nondegeneracy assumption – makes at most $m\Delta$ iterations before solving the linear feasibility problem. Because the fulfillment of the nondegeneracy assumption cannot be known a priori, we introduced a primal-dual recursive version of the algorithm that is capable of solving general degenerate problems without applying any general index selection rule (as minimal index, lexicographic ordering of variables, etc).

A geometric interpretation of our algorithm is the following. Like the criss-cross methods, the algorithm generates neighbouring points defined by the intersection of the constraints – not necessarily feasible basis – of the problem. If in any iteration the generated point lies on the feasible side of any defining hyperplane, then any further point generated will lie on the same side of the hyperplane.

In Section 2, we fix the definitions and notations used.

[1] The paper of Klee and Minty [11] ended the heroic age of the new pivot algorithms. The new pivot algorithms published afterwards were summarized by Terlaky and Zhang in [15].

In section 3, we define the MBU type simplex method, and prove some of its important properties. In case of not locally strongly degenerate pivot sequences, we calculate an upper bound for the complexity of the algorithm. In section 4, the finiteness of the algorithm is proved without any nondegeneracy assumption. Conclusions close the paper.

2 Definitions and notations

We use the following notations throughout the paper. Scalars and indices are denoted by small Latin letters, vectors by small boldface Latin letters, matrices by capital Latin letters, and finally index sets by capital calligraphic letters.

For an index set $\mathcal{G} \subseteq \mathcal{I} := \{1, 2, \ldots, n\}$ and a vector $\mathbf{v} \in \mathbb{R}^n$, we denote the subvector of $\mathbf{v}$ indexed by $\mathcal{G}$ by $\mathbf{v}_\mathcal{G}$, in other words $(\mathbf{v}_\mathcal{G})_i := v_i$, $i \in \mathcal{G}$. For any matrix $T \in \mathbb{R}^{m\times n}$, and index sets $\mathcal{F} \subseteq \mathcal{J} := \{1, \ldots, m\}$ and $\mathcal{G} \subseteq \mathcal{I}$, we denote the submatrix of T whose rows and columns are indexed by $\mathcal{F}$ and $\mathcal{G}$, respectively, by $T_{\mathcal{F}\mathcal{G}}$.

Let $B \in \mathbb{R}^{m\times m}$ be a nonsingular rectangular submatrix of $A \in \mathbb{R}^{m\times n}$ and N be the matrix consisting of the columns of A not in B. The matrix B is called a basis of the problem while the index sets $\mathcal{I}_B \subset \mathcal{I}$ and $\mathcal{I}_N \subset \mathcal{I}$ denote the index sets of the basic and nonbasic variables, respectively. It is well known that the corresponding basic solution is given by the formula $\bar{\mathbf{b}} := \bar{\mathbf{x}}_{\mathcal{I}_B} := B^{-1}\mathbf{b}$, and $T = B^{-1}N \in \mathbb{R}^{m\times(n-m)}$ defines the short pivot tableau corresponding to the basis B. The vector of nonbasic variables is $\bar{\mathbf{x}}_{\mathcal{I}_N} := \mathbf{0}$.

For a given basis B, we use the following partition:

$$\mathcal{I}_B = \mathcal{I}_B^+ \cup \mathcal{I}_B^0 \cup \mathcal{I}_B^-,$$

where

$$\mathcal{I}_B^+ = \{i \in \mathcal{I}_B : \bar{x}_i > 0\}\,,\ \mathcal{I}_B^0 = \{i \in \mathcal{I}_B : \bar{x}_i = 0\} \text{ and } \mathcal{I}_B^- = \{i \in \mathcal{I}_B : \bar{x}_i < 0\}\,.$$

If we want to emphasize the feasibility of a variable $\bar{x}_i$, we will use the notation $\mathcal{I}_B^\oplus = \mathcal{I}_B^+ \cup \mathcal{I}_B^0$ and say that $i \in \mathcal{I}_B^\oplus$.

A sample basic tableau according to the above partition is shown in Figure 1.

In most pivot algorithms known from the literature pivots are made only on positive elements in primal feasible rows (like the simplex algorithm), and

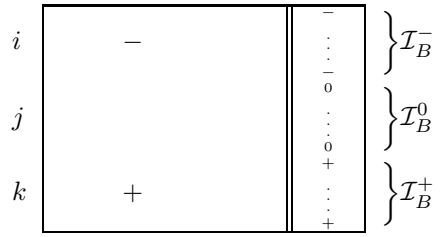

Figure 1: Partition of a basis.

only on negative elements in primal infeasible rows (like the dual simplex algorithm), or the two strategies are combined (like the criss-cross method). Recognizing this, Fukuda and Terlaky [7, 8] defined the concept of the so called admissible pivot operations. Our algorithm carries out more general pivot operations as well. A pivot on a positive value in the row of a strictly feasible variable can be viewed as the primal feasibility equivalent of the dual side admissible pivots. While handling degeneracy, our algorithm makes pivots on both negative and positive values in degenerate rows.

Definition 2.1. For a given basis B and pivot tableau T, a pivot element t_{ij} is called a *generalized admissible* pivot if

1. $i \in \mathcal{I}_B^-$ and $t_{ij} < 0$, or
2. $i \in \mathcal{I}_B^\oplus$ and $t_{ij} > 0$, or
3. $i \in \mathcal{I}_B^0$ and $t_{ij} \neq 0$.

Our algorithm uses generalized admissible pivots. Before formulating the algorithm, we need to introduce some other definitions and also refine the concept of degeneracy. Let us introduce the set

$$\mathcal{K}_s = \left\{ i \in \mathcal{I}_B^0 : t_{is} > 0 \right\}.$$

Definition 2.2. A basis B is called *degenerate* if the basic solution $\bar{\mathbf{x}}_B$ has at least one zero component, and *nondegenerate* otherwise.

The phenomenon of degeneracy is basis dependent. A basis is degenerate if and only if the right hand side vector is the linear combination of less columns of the basis than its dimension. We distinguish two kinds of degeneracy.

Definition 2.3. A degenerate basis B is called *locally weakly degenerate* with respect to index $s \in \mathcal{I}_N$ if $\mathcal{K}_s = \emptyset$, and *locally strongly degenerate* with respect to index s if $\mathcal{K}_s \neq \emptyset$.

Let us assume that for a given basis B, we have chosen the index s as a column of a generalized admissible pivot in a nondegenerate row. Observe that such a pivot that does not make any feasible variable negative, exists, if and only if $\mathcal{K}_s = \emptyset$. Such a locally weakly degenerate tableau with respect to s is shown in Figure 2.

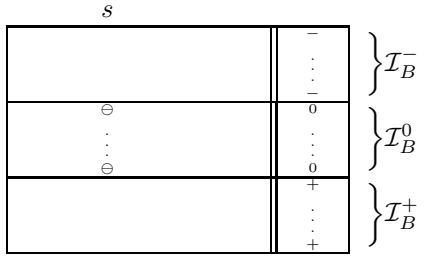

Figure 2: A locally weakly degenerate pivot tableau.

Definition 2.4. Let a basis B and an index $r \in \mathcal{I}_B^-$ be given. We call a pivot operation on t_{ij} an x_r *increasing* pivot if

1. the pivot made on t_{ij} is a generalized admissible pivot,
2. $\mathcal{I}_B^\oplus \subseteq \mathcal{I}_{B'}^\oplus$, and
3. $\hat{x}_r > \bar{x}_r$ holds,

where B' denotes the new basis, and $\hat{\mathbf{x}}$ denotes the new basic solution.

Our aim is to formulate an algorithm based on x_r increasing pivots. To our best knowledge, the only such pivot algorithm for feasibility (or general linear programming) problems known from the literature is the dual version of the algorithm of Anstreicher and Terlaky [1] (which starts from a dual feasible basic solution).

Unfortunately, there are basic tableaus where no x_r increasing pivot exists, as shown by the example of Figure 3. This problem has a feasible solution of $x_1 = x_2 = 0, x_3 = x_4 = 1$, but has no increasing pivots for the only infeasible row.

	x_3	x_4	
x_1	-1	0	-1
x_2	1	-1	0

Figure 3: Locally strongly degenerate tableau with no x_r increasing pivot, where $r = 1$.

3 The MBU type simplex algorithm for feasibility problems

In this section we formulate our MBU type simplex algorithm for solving linear feasibility problems of form (1). This algorithm has similar properties that of Anstreicher and Terlaky's MBU simplex method for linear programming problems [1]. Main idea of Anstreicher and Terlaky was that for a given dual feasible solution, the algorithm builds up the feasibility of primal variables monotonically, while in intermediate pivot steps dual feasibility is allowed to be violated but it is restored when the chosen primal driving variable achieves feasibility. Our algorithm is related to this algorithm in the sense that in our case the number of primal feasible variables increase monotonically and the pivot selection rule is similar.

In order to achieve monotonicity, our algorithm follows an intuitive path. We choose an infeasible variable in the given basic solution that we want to make feasible in the next iteration(s). So long as this variable remains infeasible, it is referred as the *driving variable*. We reduce the infeasibility of the driving variable x_r by means of x_r increasing pivots. We will show that this can always be done if the pivot tableau is nondegenerate or only locally weakly degenerate.

Our algorithm is formalized in Figure 4. In case of strongly degenerate tableaus, the algorithm is forced to use some anti-degeneracy method. A new method is proposed in section 4, to which we currently refer to as *DegProc*. Naturally, degeneracy could be handled by the classical ways known from the literature (like minimal index rule, lexicographic ordering, etc.). Our new anti-degeneracy method will ensure finiteness too.

In order to find an x_r increasing pivot, two ratio tests must be performed in the selected column. The relationship of these two ratio tests determine whether the driving variable can be made feasible in one pivot step, or x_r increasing pivot steps are necessary.

MBU type simplex algorithm for feasibility problems

Begin
Input data: $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, B$.
$T := B^{-1}A, \bar{\mathbf{b}} := B^{-1}\mathbf{b}, \quad \mathcal{I}_B^- := \{i \in \mathcal{J} \mid \bar{b}_i < 0\}$.
While $(\mathcal{I}_B^- \neq \emptyset)$ **do**
Let $r \in \mathcal{I}_B^-$ be arbitrary (driving variabe), $\quad rDone := false$.
While $(rDone = false)$ **do**
$\mathcal{J}_r^- := \{j \in \mathcal{I}_N \mid t_{rj} < 0\}$.
If $(\mathcal{J}_r^- = \emptyset)$ **then**
no feasible solution exists, **Return**
Endif
Let $s \in \mathcal{J}_r^-$ be arbitrary, $\quad \mathcal{K}_s := \{i \in \mathcal{I}_B^0 \mid t_{is} > 0\}$
If $(\mathcal{K}_s \neq \emptyset)$ **then**
$(T, l) = DegProc(T, \mathcal{I}_B^0, r)$.
If $(l \in \mathcal{I}_N)$ **then**
$s := l$.
else
no feasible soution exists **Return**
Endif
Endif
$\Theta_1 := \frac{\bar{b}_r}{t_{rs}}, \quad \Theta_2 := \min\left\{\frac{\bar{b}_k}{t_{ks}} \mid k \in \mathcal{I}_B^{\oplus}, t_{ks} > 0\right\}$.
If $(\Theta_1 \leq \Theta_2)$ **then**
pivot on $t_{rs}, \quad rDone = true$.
else
$q := \arg\min\left\{\frac{\bar{b}_k}{t_{ks}} \mid k \in \mathcal{I}_B^+, t_{ks} > 0\right\}, \quad$ pivot on t_{qs}.
Endif
Endwhile
$\mathcal{I}_B^- := \{i \mid \bar{b}_i < 0\}$
Endwhile
$\bar{\mathbf{x}}$ is a feasible solution
End

Figure 4: The MBU type simplex algorithm

The value of the two ratio tests are denoted by Θ_1 and Θ_2. From their definition, it is easy to see that $\Theta_1 > 0$ and $\Theta_2 \geq 0$. Furthermore, if the basis is not locally strongly degenerate, then $\Theta_2 > 0$ holds. The aim of the inner cycle of the algorithm is to make the driving variable feasible.

We prove that if the given basis B is not locally strongly degenerate, then the algorithm makes only x_r increasing pivots. First we investigate the case when the driving variable leaves the basis.

Proposition 3.1. For a given basis B, let $r \in \mathcal{I}_B^-$ and $q \in \mathcal{I}_B^\oplus$, $t_{qs} > 0$. Suppose that

$$\frac{\bar{b}_q}{t_{qs}} = \theta_2 \geq \theta_1 = \frac{\bar{b}_r}{t_{rs}} > 0,$$

where $\bar{b}_r < 0$, $t_{rs} < 0$, $\bar{b}_q \geq 0$ and $t_{qs} > 0$. In this case a pivot is carried out on t_{rs}. Let us denote the new basis by B', then

$$\mathcal{I}_B^\oplus \cup \{s\} \subset \mathcal{I}_{B'}^\oplus$$

holds, thus

$$\left|\mathcal{I}_{B'}^\oplus\right| > \left|\mathcal{I}_B^\oplus\right|.$$

Proof. When t_{rs} is the pivot position, the variable x_r leaves the basis, while variable x_s enters it. Let us denote the new basic solution (thus the new right hand side of the pivot tableau) by $\mathbf{b}^+$. We distinguish the following cases.

a) For the index s the right hand side becomes $b_s^+ = \frac{\bar{b}_r}{t_{rs}} = \theta_1 > 0$, making the driving variable $x_r^+ = 0$ feasible.

b) For $i \in \mathcal{I}_B^+$, $i \neq s$ holds that $b_i^+ = \bar{b}_i - \frac{t_{is}\bar{b}_r}{t_{rs}}$. If $t_{is} \leq 0$ then by using $\bar{b}_i \geq 0$, $\bar{b}_r < 0$ and $t_{rs} < 0$ we have $b_i^+ > 0$, because we add a nonnegative number to an already positive $\bar{b}_i$. Otherwise if $t_{is} > 0$ then by $b_i^+ = t_{is}\left(\frac{\bar{b}_i}{t_{is}} - \frac{\bar{b}_r}{t_{rs}}\right)$ using the condition $\frac{\bar{b}_i}{t_{is}} \geq \theta_2 \geq \theta_1 = \frac{\bar{b}_r}{t_{rs}} > 0$ we get that $b_i^+ > 0$.

c) For $i \in \mathcal{I}_B^0$ we have $b_i^+ = \bar{b}_i - \frac{t_{is}\bar{b}_r}{t_{rs}} \geq 0$, and by $\theta_2 > 0$ either $I_B^0 = \emptyset$ or $t_{is} \leq 0$, thus $b_i^+ = -\frac{t_{is}\bar{b}_r}{t_{rs}} \geq 0$.

As we have seen, no feasible variable becomes infeasible while at least the leaving basic variable x_r becomes feasible. It follows that $\left|I_{B'}^\oplus\right| > \left|I_B^\oplus\right|$. □

In the next proposition we investigate the case when a pivot is made outside the row of the driving variable. If the basis is not locally strongly degenerate for the entering nonbasic variable, the pivot made by the algorithm is still x_r increasing.

Proposition 3.2. In a given iteration of the algorithm, for the actual basis B let $r \in \mathcal{I}_B^-$ and $q \in \mathcal{I}_B^\oplus$, $t_{qs} > 0$. Suppose that

$$0 \leq \frac{\bar{b}_q}{t_{qs}} = \theta_2 < \theta_1 = \frac{\bar{b}_r}{t_{rs}},$$

where $\bar{b}_r < 0$, $t_{rs} < 0$, $\bar{b}_q \geq 0$ and $t_{qs} > 0$. In this case a pivot is carried out on t_{qs}. Let us denote the new basis by B'. Then $\mathcal{I}_B^\oplus \backslash \{q\} \subseteq \mathcal{I}_{B'}^\oplus \backslash \{s\}$ and $0 > b_r^+ \geq \bar{b}_r$. Furthermore, if $\theta_2 > 0$, then $0 > b_r^+ > \bar{b}_r$ holds.

Proof. Using the notations introduced in the previous lemma, we see that because of the ratio test, for any index $i \in \mathcal{I}_B^\oplus$ we have $i \in \mathcal{I}_{B'}^\oplus$ if $i \neq q$, as proved in the previous lemma.

Furthermore, $b_s^+ = \frac{\bar{b}_q}{t_{qs}} \geq 0$, so $s \in I_{B'}^\oplus$ thus

$$\mathcal{I}_B^\oplus \backslash \{q\} \subseteq \mathcal{I}_{B'}^\oplus \backslash \{s\}$$

providing that an already feasible variable does not become infeasible. For the index of the leading variable $b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}}$, where $-\frac{t_{rs}\bar{b}_q}{t_{qs}} \geq 0$, using that $t_{rs} < 0$, $t_{qs} > 0$ and $\bar{b}_q \geq 0$. By the condition $\theta_2 < \theta_1$ we have $0 \geq \frac{t_{rs}\bar{b}_q}{t_{qs}} > \bar{b}_r$ thus

$$0 > b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}} \geq \bar{b}_r.$$

If the basis is nondegenerate, or locally weakly degenerate, then $\theta_2 > 0$ holds by definition, so $\frac{\bar{b}_q}{t_{qs}} > 0$, thus $-t_{rs}\frac{\bar{b}_q}{t_{qs}} > 0$, and

$$0 > b_r^+ = \bar{b}_r - t_{rs}\frac{\bar{b}_q}{t_{qs}} > \bar{b}_r,$$

finalizing the proof. □

Geometrically, Proposition 3.2 can be interpreted as the new solution is closer to the constraint of the driving variable.

Summarizing Propositions 3.1 and 3.2 we obtain the following result.

Corolarry 3.1. If the MBU type simplex algorithm performs only not strongly degenerate pivots, then the algorithm makes only x_r increasing pivots, and thus it is finite.

Proof. The number of different bases is finite, therefore it suffices to prove that the algorithm is not cycling, or in other words, that a basis may not occur twice. Since we assumed that the algorithm does not visit strongly degenerate bases, it follows from propositions 3.1 and 3.2 that in each iteration, the algorithm makes x_r increasing pivots. In each step a new variable becomes feasible, or the value of the driving variable increases, thus the same basis may not return. □

Proposition 3.1 and 3.2 are results explaining the properties of the MBU type simplex algorithm. Anstreicher and Terlaky in their paper [1] proved similar results for their primal algorithm for linear programming problems.

In the next section we give a lower bound on the increment of the value of the driving variable, and consequently we provide an upper bound on the iteration number of the algorithm.

To the best of our knowledge, there is no other pivot algorithm known in the literature that has a complexity bound similar to the one we obtained from the analysis of the algorithm. General pivot algorithms share the common property that their worst case analysis are known to be exponential, or it is conjectured that there exists a problem on which the algorithm has exponential running time.

The simplex and the primal MBU simplex algorithms for solving linear optimization problems can be analyzed in a similar way to what we show in the next section. While the geometric interpretation of the primal MBU simplex algorithm is that we approach a selected infeasible constraint, the phase-1 simplex algorithm tries to decrease the sum of infeasibilities.

3.1 Complexity bound for not locally strongly degenerate pivot tableau sequences

In this subsection we assume that our algorithm visits only not locally strongly degenerate tableaus. Degeneracy is handled in section 4.

By the definition of the pivot tableau and the basic solution, $\mathbf{t}_s = B^{-1}\mathbf{a}_s$ and $\bar{\mathbf{b}} = B^{-1}\mathbf{b}$; thus the vectors $\mathbf{t}_s$ and $\bar{\mathbf{b}}$ can be considered as the unique

solutions of the linear equations $B\mathbf{u} = \mathbf{a}_s$ and $B\mathbf{v} = \mathbf{b}$. For any index $i \in I_B$ the Cramer's rule yields

$$t_{is} = \frac{\det(B_{is})}{\det(B)} \quad \text{and} \quad \bar{b}_i = \frac{\det(B_i)}{\det(B)},$$

where the matrix $B_{is} \in \mathbb{R}^{m\times m}$ is the modification of the regular basic matrix B such that the i^{th} column is replaced by vector $\mathbf{a}_s$, and similarly the matrix B_i is obtained from B by replacing the i^{th} column by vector $\mathbf{b}$. For an x_r increasing pivot that does not make the driving variable feasible, as seen in Proposition 3.2, we have

$$b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}} = \frac{\det(B_r)}{\det(B)} - \frac{\frac{\det(B_{rs})}{\det(B)}\frac{\det(B_q)}{\det(B)}}{\frac{\det(B_{qs})}{\det(B)}} = \frac{\det(B_r)}{\det(B)} - \frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)},$$

where

$$-\frac{\det(B_{rs})}{\det(B_{qs})}\frac{\det(B_q)}{\det(B)} > 0$$

holds by the fact that the basis is not locally strongly degenerate. Let

$$\Delta_A := \min\left\{ -\frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)} : \begin{array}{l} B \text{ is a regular submatrix of A, and} \\ \frac{\det(Brs)}{\det(B)} < 0,\ \frac{\det(B_q)}{\det(B)} > 0,\ \frac{\det(B_{qs})}{\det(B)} > 0 \end{array} \right\}$$

is the minimal increase of the driving variable's value. Assuming that in all pivot transformations of the tableau $\theta_2 > 0$ holds, we have that $\Delta_A > 0$ is a finite number and

$$b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}} = \frac{\det(B_r)}{\det(B)} - \frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)} \geq \frac{\det(B_r)}{\det(B)} + \Delta_A$$

thus an x_r increasing pivot either makes the leading variable feasible, or increases its value by at least Δ_A. We now bound the maximum absolute value that an infeasible variable can take during the algorithm. Let

$$\Delta_{\max} := \max\left\{ -\frac{\det(B_r)}{\det(B)} : \begin{array}{l} sgn(\det(B_r)) = -sgn(\det(B)), \\ B \in \mathbb{R}^{m\times m} \text{ is a regular submatrix of } A \end{array} \right\}$$

be maximal RHS value calculated with the help of the Cramer's rule. If there is any basis for which there is a negative right hand side value, then the number $\Delta_{\max}$ is positive and finite. Let $\Delta \in \mathbb{Z}$ be such that $\Delta = \left\lceil \frac{\Delta_{\max}}{\Delta_A} \right\rceil$, thus $\Delta \in \mathbb{N}$.

We are ready to bound the number of pivots necessary to make the driving variable feasible.

Proposition 3.3. Assume that the algorithm visits only not locally strongly degenerate pivot tableaus. Let $r \in I_{\overline{B}}$ be the index of the driving variable. There can be at most Δ pivot operations before the driving variable becomes feasible.

Proof. By the definition, the value of the driving variable cannot be smaller than $-\Delta_{\max}$. The value of the driving variable increases by at least Δ_A in every iteration, thus there cannot be more than Δ iterations before the next x_r increasing pivot makes the driving variable feasible. □

We are now ready to prove the bound on the complexity of the algorithm.

Theorem 3.1. *Consider the feasibility problem (1). Assume that the MBU type pivot algorithm visits only not locally strongly degenerate pivot tableaus in solving (1). Then the algorithm is finite, and there can be at most $m\Delta$ pivots.*

Proof. By proposition 3.3, there can be at most Δ pivot operations before the algorithm sets feasibility in the row of the driving variable or proves infeasibility. The number of driving variables during the algorithm is bounded by the number of rows, for any variable and any row of the pivot tableau that become feasible remains so by Propositions 3.1 and 3.2, thus the algorithm may not cycle, and there can be at most $m\Delta$ pivots before solving the problem, or proving that it is infeasible. □

We have proved under the nondegeneracy assumption that the algorithm is finite, and we can bound the required number of pivot operations. This upper bound is generally not tight, and it would be interesting to find such problem classes where this bound can easily be determined, that is, the values of Δ_A and $\Delta_{\max}$ are easily computable. A naturally arising problem class would be for which the matrix A is totally unimodular. Most of those problems coming from combinatorial optimization are highly degenerate, thus the complexity estimate would only bound the number of not locally strongly degenerate pivot steps. In case of $\mathbf{b} \in \mathbb{Z}^m$, using simple calculations and expansion of the corresponding determinant using the column of $\mathbf{b}$, we get $\Delta \leq \|\mathbf{b}\|_1$. It would also be interesting to find proper perturbations for given problem classes to handle strong degeneracy. Remark that the perturbation techniques known from the literature (like the ϵ-perturbation technic) does not give an appropriate answer to the problem, since they usually drastically effect the values of Δ_A and $\Delta_{\max}$ as well.

4 Handling strong degeneracy

In this section, we discuss the *DegProc* of the MBU type algorithm presented in Figure 4.

Algorithmically, degeneracy is often handled by perturbation or index selection rules (lexicographic, minimal index). This is also possible in case of the MBU type simplex algorithm, but we follow a different approach to handle problems caused by degeneracy.

The key issue of the analysis presented in the previous section was that the tableaus visited by the algorithm were all not locally strongly degenerate. A procedure for handling degeneracy makes such pivots which are based on the row of the driving variable and the degenerate submatrix (consisting of all degenerate rows) that do not change the current basic solution, but transform the basic tableau such that it either becomes primal infeasible, or there will be at least one column that have negative entry in the row of the driving variable, and moreover is locally weakly degenerate. While solving the primal (dual) degenerate subproblem, we only make pivots in the degenerate rows, and for solving the subproblems we use the dual (primal) versions of the MBU type simplex algorithms, thus the solution process of the subproblems carry the same already shown basic properties over the iterations as the MBU type simplex algorithm.

Figure 5. summarizes the key pivot tableaus of the algorithm.

Observe that the infeasible tableau of the original primal problem has the same structure as the feasible tableau of problem $\text{Dual}_\oplus$ except for the column of $\mathbf{b}$. The structure of the infeasible tableau of problem $\text{Dual}_\oplus$ is the same as for the locally weakly degenerate tableau of the original primal problem. Problem $\text{Primal}_\ominus$ is the negative signed equivalent of problem (1). The importance of this problem is that it has a similar, but mutual relationship between problems $\text{Dual}_\oplus$ and $\text{Primal}_\ominus$.

The procedure for degeneracy follows the idea explained above: if the tableau is strongly degenerate with respect to the selected nonbasic variable, it tries to show that the problem is infeasible. To prove this, on the degenerate subproblem – consisting of the degenerate rows in the primal case, and of the degenerate columns in the dual case – as pivot tableau, and the row of the driving variable as dual right hand side, it solves a dual subproblem, but carries out the pivot operations on the whole tableau. If the dual subproblem could be solved, that proves the infeasibility of the original problem, while if the subproblem had no solution, that provides a locally weakly degenerate tableau with respect to a proper nonbasic variable for the original problem.

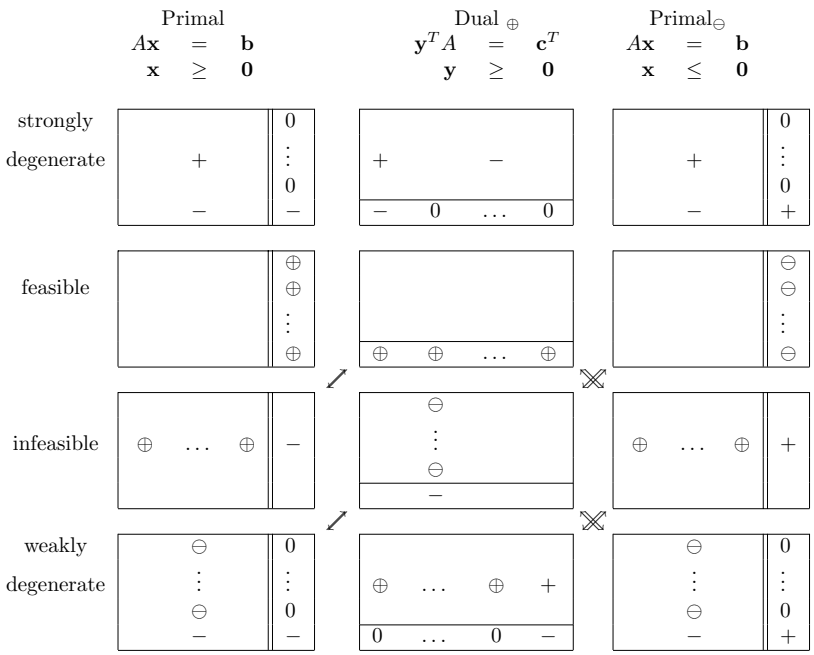

Figure 5: Key tableaus of the algorithm

We mention that the dual subproblem could naturally be solved by arbitrary pivot methods, but we prefer the dual version of the MBU type simplex algorithm.

We prove that while solving the degenerate subproblem, the actual basic solution of the algorithm does not change.

Proposition 4.1. Let a degenerate pivot tableau $T \in \mathbb{R}^{m\times n}$ be given, and denote the index set of degenerate rows by $\mathcal{D} = \mathcal{I}_B^0$. Then any pivot made on the elements of submatrix $T_{\mathcal{D}\mathcal{I}}$ does not change the current basic solution.

Proof. Let the chosen generalized admissible pivot position be $t_{ij} \in T_{\mathcal{D}\mathcal{I}}$. We show that the right-hand side of the tableau does not change after the pivot, namely $b_i^+ = \frac{b_i}{t_{ij}} = 0 = b_i$ and $b_k^+ = b_k + t_{kj}\frac{b_i}{t_{ij}} = b_k + 0 = b_k$, where $k \neq i$. □

Because of the nature of the procedure handling degeneracy, it is easier to

formulate it as a recursive method. The pseudo-codes of the sub-procedures are summarized in Figures 6. and 7. The sub-procedures makes pivots on the whole tableaus, but consider only the sub-tableau defined by the index sets $\mathcal{F}$ and $\mathcal{G}$. The column indexed by b, and row indexed by c play the roles of primal and dual right-hand sides, respectively.

$(T, l) =$ **DualSubMBU**$(T, c, \mathcal{F}, \mathcal{G})$
  While $((\mathcal{J}_B^- := \{j \in \mathcal{G} : t_{cj} < 0\}) \neq \emptyset)$ **do**
    Let $r \in \mathcal{J}_B^-$ be arbitrary (driving variable), $rDone := false$.
    While $(rDone = false)$ **do**
      If $((\mathcal{I}_r^+ := \{i \in \mathcal{F} \mid t_{ir} > 0\}) = \emptyset)$ **then**
        The calling pivot tableau is weakly degenerate, **Return**(T, r).
      Endif
      Let $s \in \mathcal{I}_r^+$, $\mathcal{K}_s = \{k \in \mathcal{G} \mid t_{ck} = 0, t_{sk} < 0\}$.
      If $(\mathcal{K}_s \neq \emptyset)$ **then** (when subtableau $T_{\mathcal{F}\mathcal{G}}$ is strongly degenerate)
        $(T, s) :=$ PrimalSubMBU$(T, r, \mathcal{F}, \{j \in \mathcal{G} \mid t_{cj} = 0\})$.
        If $(s = -1)$ **then**
          The calling tableau is weakly degenerate, **Return**(T, r).
        Endif
      Endif
      $\theta_2 := \max\left\{\frac{t_{ck}}{t_{sr}} \mid k \in \mathcal{G}, t_{ck} > 0, t_{sk} < 0\right\}$
      If $\left(\left(\theta_1 := \frac{t_{cr}}{t_{sr}}\right) \leq \theta_2\right)$ **then**
        pivot on t_{sr}, $rDone = true$.
      else
        $q := \arg\max\left\{\frac{t_{ck}}{t_{sr}} \mid k \in \mathcal{G}, t_{ck} > 0, t_{sk} < 0\right\}$.
        pivot on t_{sq}.
      Endif
    Endwhile
  Endwhile
  The calling tableau is infeasible, **Return**$(T, -1)$.
End

Figure 6: The dual subprocedure starts with the solution of a subproblem of the form Dual$_\oplus$.

Because the primal subproblems called by *DegProc* try to achieve a right-hand side vector of different sign as the original algorithm, in what follows

we call a primal variable feasible if its sign is adequate for the corresponding feasibility problem or zero, and infeasible otherwise. For clear expression, in the discussion of the subproblems, we will say only values instead of variables when referring the right hand side values, because only for the original problem does variables actually correspond to right hand side values.

The analysis of the procedure for solving the dual subproblems is completely analogous to the primal version, and is left to the reader. Using Figure 5. the concept of dual side locally weak and strong degeneracy, as well as concept of the x_r increasing pivot can be defined similarly as in 2.2., 2.3. and 2.4.

We show that primal and dual subalgorithms have similar properties as the primal algorithm working on nondegenerate problems.

Proposition 4.2. For the PrimalSubMBU and DualSubMBU procedures, if the corresponding tableau is not locally strongly degenerate, then the pivot steps carried out are increasing pivots for the column indexed by b or for the row indexed by c, respectively.

Proof. The first part of the proposition follows immediately from Proposition 3.1. while the part corresponding to the dual subalgorithm can be proved similarly. □

Although we have already stated the main idea behind the recursive algorithm, we now formalize it for both subprocedures.

Proposition 4.3. Suppose that the algorithm is started from problem P_0, then because of repeated locally strongly degenerate tableaus, the $P_1 := SubDual(T_1, k_1, \mathcal{F}_1, \mathcal{G}_1)$, $P_2 := SubPrimal(T_2, k_2, \mathcal{F}_2, \mathcal{G}_2), \ldots,$ $P_l := SubPrimal(T_l, k_l, \mathcal{F}_l, \mathcal{G}_l)$ (or $P_l := SubDual(T_l, k_l, \mathcal{F}_l, \mathcal{G}_l)$) recursive calling sequence occurs. Then the pivots carried out while solving subproblem P_l does not change any right hand side of problems P_i, where $(i = 1, \ldots, l-1)$ and $k_i \notin \mathcal{F}_l \cup \mathcal{G}_l$.

Proof. The recursive steps involve only degenerate rows and columns, thus by Proposition 4.1 our statement holds. □

The procedure starting the recursion and handling degeneracy can easily be formulated as shown in Figure 8.

The relationship of the different subproblems are shown in Figure 9. The figure shows a possible primal-dual-primal calling sequence. Phrases *SubPrimal* and *SubDual* refer to the type of the subproblem. The structure

$(T, l) =$ **PrimalSubMBU**$(T, b, \mathcal{F}, \mathcal{G})$
  While $((\mathcal{I}_B^+ := \{i \in \mathcal{F} : t_{ib} > 0\}) \neq \emptyset)$ **do**
    Let $r \in \mathcal{I}_B^+$ be arbitrary (driving variable), $rDone := false$.
    While $(rDone = false)$ **do**
      If $((\mathcal{J}_r^- := \{j \in \mathcal{G} \mid t_{rj} < 0\}) = \emptyset)$ **then**
        The calling tableau is weakly degenerate, **Return**(T, r).
      Endif
      Let $s \in \mathcal{J}_r^-$, $\mathcal{K}_s := \{k \in \mathcal{F} : t_{kb} = 0, t_{ks} > 0\}$.
      If $(\mathcal{K}_s \neq \emptyset)$ **then** (when subtableau $T_{\mathcal{FG}}$ is strongly degenerate)
        $(T, s) :=$ DualSubMBU$(T, r, \{i \in \mathcal{F} \mid t_{ib} = 0\}, \mathcal{G})$.
        If $(s = -1)$ **then**
          The calling tableau is weakly degenerate, **Return**(T, r).
        Endif
      Endif
      $\theta_2 := \min\left\{\frac{|t_{kb}|}{t_{ks}} \mid k \in \mathcal{F}, t_{kb} < 0, t_{ks} > 0\right\}$.
      If $\left(\left(\theta_1 := \frac{|t_{rb}|}{t_{rs}}\right) \leq \theta_2\right)$ **then**
        pivot on t_{rs}, $rDone = true$
      else
        $q := \arg\min\left\{\frac{|t_{kb}|}{t_{ks}} \mid k \in \mathcal{F}, t_{kb} < 0, t_{ks} > 0\right\}$.
        pivot on t_{qs}.
      Endif
    Endwhile
  Endwhile
  The calling tableau is infeasible, **Return**$(T, -1)$.
End

Figure 7: The *PrimalSubMBU* starts with the solution of a subproblem of form Primal$_\ominus$.

of the basic tableaus corresponding to locally strongly degenerate bases has already been presented in Figure 5.

We now prove the finiteness of the MBU algorithm without any nondegeneracy assumption.

Theorem 4.1. *The MBU type simplex algorithm for any feasibility problems is finite.*

(T, l) =**DegProc**$T, \mathcal{I}_B^0, r$**)**
  (T, l)=(PrimalSubMBU$(T, \mathcal{I}_B^0, \{1, \ldots, n\}, n+1)$).
  Return(T, l).
End

Figure 8: A possible anti degeneracy procedure, DegProc.

Proof. While the algorithm visits only not locally strongly degenerate problems, the algorithm carries out x_r increasing pivots according to 4.2, thus the same basis may not return. Then the algorithm may not cycle. In case the corresponding pivot tableau is locally strongly degenerate for a choice of a nonbasic variable, the algorithm calls the PrimalSubMBU or DualSubMBU subprocedures for strictly smaller problems, thus the depth of recursion must be no greater than $2m \leq n + m$.

Consider the case when the recursively called *DualSubMBU* solves the subproblem. If it stops with an infeasible sub-tableau, then the corresponding calling (sub)problem become locally weakly degenerate to the proper nonbasic variable, thus the procedures continues with an increasing pivot. If the *DualSubMBU* stopped with a feasible sub-tableau, then the primal (sub)problem above is infeasible. This means the infeasibility of the original problem if the calling procedure was the *DegProc* procedure. Otherwise, the dual subproblem one more step above become locally weakly degenerate, thus it continues with an increasing pivot.

Similar connections hold when the *PrimalSubMBU* solves the corresponding subproblem.

Because the depth of recursion is bounded, and the returning sub-procedures provide possibility to an increasing pivot for the calling procedure, no basis may occur twice. The number of different bases is finite, thus the algorithm is finite. □

Observe that both the algorithm and its recursive subalgorithms make increasing pivots to the corresponding (sub)problem and use recursion. Thus it is possible to generalize the complexity bound of the not locally strongly degenerate case, but because of the recursion the implied bound greatly depends on number of degenerate subproblem calls.

Although the analysis presented in Section 3.1. can be carried out for the first phase of the simplex algorithm, the presented anti cycling recursion

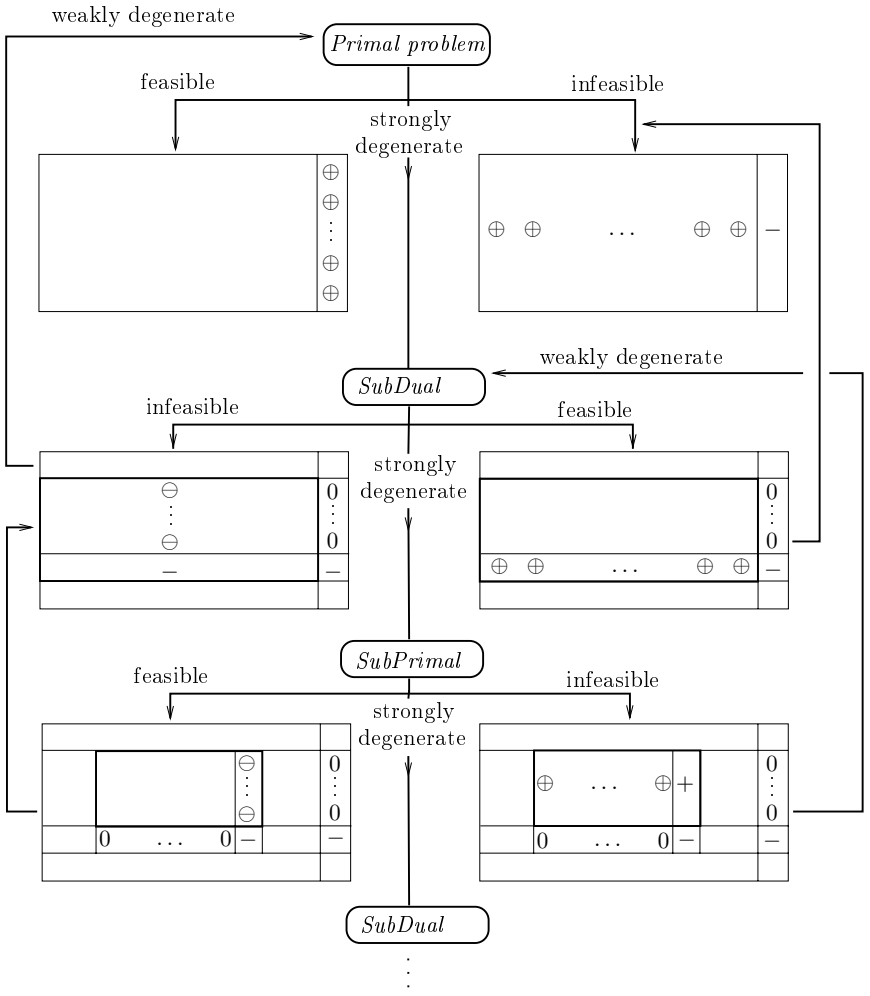

Figure 9: A Primal-Dual-Primal sequence of subproblems.

procedure cannot be naturally applied for the simplex algorithm. The main reason for this is that when a subproblem is feasible, we greatly make use of the infeasibility of the driving variable to reach the conclusion that the calling problem is infeasible.

4.1 Algorithm variants

As already stated, it would be possible to solve the degenerate subproblems with an arbitrary pivot method, similarly like in the case of the Hungarian method for linear programming [10], in which the criss-cross method was used [14].

In practice, the efficiency of the algorithm could be increased if the choice of the driving variable as well as the choice of the variable to enter the basis would not be arbitrary, although this freedom may help in case of numerically bad problems.

The most expensive operation of the pivot methods is the calculation of the updated pivot row and column. However, for an efficient choice of pivot position, it could be interesting to look for locally weakly degenerate columns for the driving variable. Similarly, it may happen that on a pivot tableau there are infeasible variables that could be made feasible in only one step, but another one would require even recursion.

If we know a greater part of the pivot tableau, it may be worth to check for possible change of the driving variable. If in any step the pivot tableau is locally strongly degenerate for the selected driving variable, it may as well happen that there is an infeasible variable that could be made feasible while preserving the monotonicity of the cardinality of the feasible variables. Such a pivot would preserve the finiteness of the algorithm.

It must be noted however, that unfortunately in some cases neither of these strategies can be used, and recursion is necessary, like in the example of Figure 3.

5 Conclusion

We introduced a new MBU type simplex algorithm for linear feasibility problems, and presented a new anti-degeneracy procedure. The same analysis can be made for the first phase of the primal simplex algorithm, and the original MBU simplex algorithm under the assumption that the tableaus visited are not strongly degenerate. The presented anti-degeneracy procedure uses

the special structure of the MBU type simplex algorithm, therefore is not applicable for the simplex method.

The anti-degeneracy rule is new, and differs from the known methods from the literature. The new complexity analysis of the algorithm proves to be more interesting for locally weakly degenerate pivot sequences, and is very sensitive to perturbation. It would be interesting to find perturbations that fit to the presented anti-degeneracy method.

References

[1] Anstreicher, K. M. and Terlaky T., *A Monotonic Build-Up Simplex Algorithm for Linear Programming*. Operations Research 42:556-561, 1994.

[2] Borgwardt, K. H., The Simplex Method: A Probabilistic Analysis. Algorithms and Combinatorics, Vol. 1, Springer, Berlin, 1987.

[3] Dantzig, G.B., Programming in a Linear Structure. Comptroller, USAF, Washington D.C., 1948.

[4] Dantzig, G.B., Linear Programming and Extensions. Princeton University Press, Princeton, New Jersey, 1963.

[5] Farkas Gy., *A Fourier-féle mechanikai elv alkalmazásai.* Mathematikai és Természettudományi Értesítő, 12:457-472, 1894.

[6] Farkas Gy., *A Fourier-féle mechanikai elv alkalmazásainak algebrai alapjáról.* Mathematikai és Physikai Lapok 5:49-54, 1896.

[7] Fukuda K. and Terlaky T., *Criss-cross methods: A fresh view on pivot algorithms*. Mathematical Programming 79:369-395, 1997.

[8] Fukuda K. and Terlaky T., *On the existence of a short admissible pivot sequence for feasibility and linear optimization problems*. Pure Mathematics with Applications 10 no. 4, 431-447, 1999.

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

[10] Klaszky, E., Terlaky, T., *Magyar módszer típusú algoritmusok lineéris programozási feladatok megoldására*. Alkalmazott Matematikai Lapok 12, 1-14, 1986.

[11] Klee, V. and Minty, G.J., *How good is the simplex algorithm?* In O. Shisha, editor, *Inequalities III, Academic Press*, New York, 1972.

[12] Maros I., Computational Techniques of the Simplex Method. *Kluwer Academic Publishers*, Boston, 2003.

[13] Prékopa A., *On the development of optimization.* American Mathematical Monthly, 87:527–542, 1980.

[14] Terlaky, T., *A convergent criss-cross method*, Mathematische Operationsforschung und Statistik Series Optimization , Vol. 16, No. 5, 683-690, 1985.

[15] Terlaky T. and Zhang S., *Pivot rules for linear programming: A survey on recent theoretical developments.* Annals of Operations Research 46:203-233, 1993.

[16] Todd, M. J., *Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems.* Mathematical Programming 35:173-192, 1986.

Zsolt Csizmadia, Tibor Illés
Eötvös Loránd University, Department of Operations Research
H- 1117 Budapest, Pázmány Péter sétány 1/C.
E-mail: csisza@math.elte.hu, illes@math.elte.hu

Filiz Bilen
Eastern Mediterranean University, Department of Mathematics
Famagusta, North-Cyprus
E-mail: filiz.bilen@emu.edu.tr

Operations Research Reports No. 2005-03

Earlier Research Reports

1991-01 T. Illés, J. Mayer and T. Terlaky: A new approach to the colour matching problem

1991-02 E. Klaszky, J. Mayer and T. Terlaky: A geometric programming approach to the chanel capacity problem

1992-01 Edvi T.: Karmarkar projektív skálázási algoritmusa

1992-02 Kassay G.: Minimax tételek és alkalmazásaik

1992-03 T. Illés, I. Joó and G. Kassay: On a nonconvex Farkas theorem and its applications in optimization theory

1992-04 Interior point methods. Proceedings of the IPM 93. Workshop jan. 5. 1993

Recent Operations Research Reports

2003-01 Zsolt Csizmadia and Tibor Illés: New criss-cross type algorithms for linear complementarity problems with sufficient matrices

2003-02 Tibor Illés and Ádám B. Nagy: A sufficient optimality criteria for linearly constrained, separable concave minimization problems

2004-01 Tibor Illés and Marianna Nagy: The Mizuno–Todd–Ye predictor–corrector algorithm for sufficient matrix linear complementarity problem

2005-01 Miklós Ujvári: On a closedness theorem

2005-02 Faluközy Tamás és Vizvári Béla: Az árutőzsde gabona szekciójának árvárakozásai a kukorica kereskedésének tükrében

2005-03 Bilen Filiz, Zsolt Csizmadia and Tibor Illés : Anstreicher-Terlaky type monotonic simplex algorithms for linear feasibility problems