

Algorithms in Linear Programming

L. Finschi, Eidgenössische Technische Hochschule Zürich

K. Fukuda, Eidgenössische Technische Hochschule Zürich

H.-J. Lüthi, Eidgenössische Technische Hochschule Zürich

Summary: We study linear programming (LP) algorithms. Of particular interest are bounds for the number of elementary arithmetic operations necessary to solve a linear program. The best bounds that depend only on the sizes of a basis and a nonbasis have been found for a family of randomized pivoting algorithms. However, the original descriptions and analyses of these algorithms use several different geometric and abstract settings. In this paper we present a unified framework in which we describe two known algorithms as special simplex methods and analyse their complexities and differences.

1. Introduction

A linear programming (LP) problem is to find a maximizer (or minimizer) of a linear function over a system of linear inequalities. We study LP algorithms and upper bounds for the number of elementary arithmetic operations necessary to solve LP problems; for this we assume that each operation can be performed in constant time. In particular we are interested in bounds that depend only on the size m of a basis and the size n of a nonbasis. To find an LP algorithm that is polynomial in m and n — usually this is called strongly polynomial — is one of the most challenging open questions.

The simplex method of Dantzig [2] and also the algorithms of Kachiyan [9] and Karmarkar [8] do not satisfy this condition. New algorithms with complexity bounds in terms of m and n were introduced by Megiddo [11], Seidel [13], and others, but the dependence on n remained exponential. In 1992 Sharir and Welzl [14] and also Kalai [6] presented randomized algorithms which turned out — after proofs first of Kalai and then of Matoušek, Sharir, and Welzl [10] — to have a so-called subexponential expected running time. The proofs rely on similar ideas, whereas the descriptions of these algorithms use very different frameworks.

Our main motivation is to understand the Sharir-Welzl algorithm (here abbreviated by MSW) and Kalai's algorithm as pivoting algorithms so that we can consider them as refinements of the simplex method and compare them with any other pivoting algorithm on the same ground. In 1995 Goldwasser [5] showed that MSW and some variant of Kalai's algorithms are dual to each other. Kalai [7] claimed in 1997 that another variant of his algorithms and MSW are equivalent in a dual setting. This paper compares these last two algorithms in our framework and shows that their behaviors are slightly different. The main importance is not the difference we found but the simplicity and the preciseness of our framework that permit rigorous analysis and even straightforward implementations.

2. The Framework: Dictionaries and Oracles

For a rigorous treatment, we present in this section the definitions and notations which we use for the description of the algorithms in the following sections. The reader who is familiar with dictionaries

and pivot operations may directly go to Section 5.

2.1 Dictionaries and Pivot Operation

For two finite and nonempty sets R and C an $R \times C$ -matrix M consists of the *entries* M_{ij} for $i \in R$, $j \in C$. For nonempty subsets $I \subseteq R$, $J \subseteq C$ we denote the *submatrix* corresponding to the entries M_{ij} for $i \in I$, $j \in J$ by M_{IJ} . We set $M_I := M_{IC}$, $M_{.J} := M_{RJ}$, and $M_i := M_{\{i\}}$ etc.

We assume without loss of generality that an LP problem is given in the following *dictionary form*:

$$\max x_f \text{ subject to } x_{\overline{B}} = D x_{\overline{N}}, x_g = 1, x_E \geq 0, \quad (1)$$

where D is a given rational $\overline{B} \times \overline{N}$ -matrix for index sets $\overline{B} = B \cup \{f\}$ and $\overline{N} = N \cup \{g\}$, and $E = B \cup N$; x is a variable \overline{E} -vector, where $\overline{E} = \overline{B} \cup \overline{N}$. $D = D(\overline{B})$ is called the *dictionary*, \overline{B} and \overline{N} the corresponding *basis* and *nonbasis*, respectively; we set $m := |B|$ and $n := |N|$.

We consider a linear program P as given by (1). For $i \in B$, $j \in N$ holds: We can find a dictionary form of P with basis $\overline{B} \setminus \{i\} \cup \{j\}$ if and only if $D_{ij} \neq 0$. The replacement of \overline{B} by $\overline{B} \setminus \{i\} \cup \{j\}$ for $D_{ij} \neq 0$ is called *the pivot operation on (i, j)* . The pair (i, j) is called *the pivot*. The new dictionary $\tilde{D} = D(\overline{B} \setminus \{i\} \cup \{j\})$ after a pivot operation on (i, j) can be computed in time $O(mn)$.

We remark that the dictionary form of the dual linear program is

$$\max y_g \text{ subject to } y_{\overline{N}} = -D^T y_{\overline{B}}, y_f = 1, y_E \geq 0.$$

Compared with the primal dictionary form (1), basis and nonbasis variables interchange, in particular f and g interchange, and *the dual dictionary* of D is $-D^T$.

A rational \overline{E} -vector x is called a *basic solution (of P)*, if there exists a basis \overline{B} of P such that $x_N = 0$ and $x_g = 1$ (and then $x_{\overline{B}} = D_{.g}$). We call this $x = x(\overline{B})$ *the basic solution corresponding to \overline{B}* . In order to find the optimal solution we have only to visit the basic solutions: If there exists an optimal solution of P , then there also exists an optimal solution of P which is a basic solution.

For a linear program P given as in (1) and for $R \subseteq E$ the *contraction problem of P with respect to R* is defined as P with additional constraints $x_R = 0$ and denoted by P/R . If $R \subseteq N$ we can write P/R in the form (1) by substituting $D_{.N \setminus R}$ for the dictionary (i.e. we delete the columns R in the dictionary D). The value x_f of an optimal solution x of P/R is denoted by $v(P/R)$; if P/R is infeasible we set $v(P/R) := -\infty$, if P/R is unbounded we set $v(P/R) := +\infty$. We will use the obvious extension of the linear order on the real numbers to $-\infty$ and $+\infty$.

Definition 1 (Properties of a Dictionary) A dictionary $D = D(\overline{B})$ is called *feasible* if $D_{Bg} \geq 0$, *inconsistent* if exists $i \in B$ such that $D_{ig} < 0$ and $D_{iN} \leq 0$, and *degenerate* if exists $i \in B$ such that $D_{ig} = 0$. We call D *dual feasible*, *dual inconsistent*, or *dual degenerate* if the dual dictionary $-D^T$ is feasible, inconsistent, or degenerate, respectively.

The dictionary is called *optimal* if it is feasible and dual feasible, *unbounded* if it is feasible and dual inconsistent, *terminal* if it is optimal or inconsistent or dual inconsistent, and it is called *nearly optimal w.r.t. $j \in N$* if it is not optimal and if $D_{.N \setminus \{j\}}$ is optimal.

We call a linear program P *nondegenerate* if all feasible dictionaries are nondegenerate; there is an obvious dual notion of this definition. Remark the correspondences between a dictionary $D(\overline{B})$ and the basic solution $x(\overline{B})$.

Definition 2 (Simplex Pivot) Consider a feasible dictionary D . We call (i, j) a *simplex pivot* if $D_{fj} > 0$, and $D_{ij} < 0$ and if after a pivot operation on (i, j) the new dictionary \tilde{D} is feasible.

By a pivot operation on a simplex pivot (a *simplex pivot operation*) the objective value does not decrease ($D_{fg} \leq \tilde{D}_{fg}$). If D is nondegenerate, the objective value increases ($D_{fg} < \tilde{D}_{fg}$). There exists a simplex pivot (i, j) in D if and only if D is not terminal.

2.2 Oracles for Dictionaries

We will use the following *oracles* which reflect properties of the sign structure of dictionaries:

- The value of **Terminal**(D, R) is **True** if the dictionary D is terminal for the contraction problem P/R (i.e. if $D_{N \setminus R}$ is terminal), otherwise **False**.
- The value of **NearlyOptimal**(D, R) is **(True, j)** if the dictionary $D_{N \setminus R}$ is nearly optimal w.r.t. $j \in N$, otherwise **(False, \emptyset)**.
- For a feasible and not terminal dictionary D and $j \in N$ such that $D_{fj} > 0$, the value of **Pivot**(D, j) is a dictionary after an operation on a simplex pivot (i, j) in the given column (this simplex pivot is unique for nondegenerate linear programs). For input (D, j) that does not satisfy the above requirements the value of the oracle is the given dictionary D .

In a usual implementation the running time of these oracles will be $O(mn)$, $O(m + n)$, and $O(mn)$.

3. The Algorithms

3.1 Preparation: Simplex Algorithm

The algorithms we will discuss in the following are variants of the primal simplex algorithm which is reviewed here in dictionary notation. We use the oracles of Subsection 2.2.

Algorithm Simplex:

Input: A feasible dictionary D , defining a linear program P as in (1).

Output: A terminal and feasible dictionary D^* for P , i.e. D^* is either optimal or unbounded.

Simplex(D)

begin

while not **Terminal**(D, \emptyset) **do**

 choose any pivot column $j \in N$ such that $D_{fj} > 0$;

$D :=$ **Pivot**(D, j)

endwhile;

return D

end.

In the following algorithms the choice of the pivot column will be given by randomized rules based on the sign pattern of the dictionaries.

We present in this subsection an algorithm of Matoušek, Sharir, Welzl [10] and an algorithm of Kalai [7] with the main complexity theorems. The proofs are given in Section 4. The two algorithms are called here **MSW** and **Kalai** and work with a feasible starting dictionary and for nondegenerate linear programs.

We present **MSW** as a primal simplex algorithm (so dualized compared to the original algorithm):

Algorithm MSW:

Input: A feasible dictionary D defining a nondegenerate linear program P , a set $R \subseteq N$.

Output: A terminal and feasible dictionary D^* for P/R , i.e. D^* is either optimal or unbounded for P/R .

MSW(D, R)

begin

if $R = N$ **then return** D

else

 choose $e \in N \setminus R$ at random;

$\tilde{D} := \text{MSW}(D, R \cup \{e\})$;

if **Terminal**(\tilde{D}, R) **then return** \tilde{D}

else

$\hat{D} := \text{Pivot}(\tilde{D}, e)$;

return **MSW**(\hat{D}, R)

endif

endif

end.

Theorem 3 (Analysis of MSW) *The expected number of pivot operations that are generated by **MSW**(D, \emptyset) is at most $e^{4\sqrt{m \ln(n+1)}}$. The algorithm is finite and terminates with correct output.*

Here is our description of Kalai's algorithm:

Algorithm Kalai:

Input: A feasible dictionary D defining a nondegenerate linear program P , a set $R \subseteq N$.

Output: A terminal and feasible dictionary D^* for P/R , i.e. D^* is either optimal or unbounded for P/R .

Kalai(D, R)

begin

if **Terminal**(D, R) **then return** D

elseif $|N \setminus R| = 1$ (with $N \setminus R = \{e\}$) **then**

$\tilde{D} := \mathbf{Pivot}(D, e)$;

return \tilde{D}

else

$(v, \tilde{e}) := \mathbf{NearlyOptimal}(D, R)$;

if $v = \mathbf{True}$ **then**

 choose $e \in (N \setminus R) \setminus \{\tilde{e}\}$ at random;

else choose $e \in N \setminus R$ at random;

endif;

$\hat{D} := \mathbf{Kalai}(D, R \cup \{e\})$;

return $\mathbf{Kalai}(\hat{D}, R)$

endif

end.

Theorem 4 (Analysis of Kalai) *The expected number of pivot operations that are generated by $\mathbf{Kalai}(D, \emptyset)$ is for $n > 0$ at most $e^{4\sqrt{(m+n)\ln n}} + 1$; for $n = 0$ there is no pivot operation generated. The algorithm is finite and terminates with correct output.*

Both algorithms are indeed variants of the simplex algorithm: Obviously there will be some sequence of simplex pivot operations, and the algorithms will return a dictionary which is terminal for P/R if they terminate (observe that in both algorithms the argument of any **return** statement either is a dictionary which is terminal for P/R or a recursive call with the same contraction set R).

4. Discussion of the Algorithms

4.1 Analysis of MSW

We discuss in this subsection the algorithm of Matoušek, Sharir, Welzl **MSW**. The analysis is the same as in the original paper [10], but instead of the notation of so-called LP-type optimization problems we use the setting of dictionaries (see Subsection 2.1) and oracles (see Subsection 2.2).

We write $v(R)$ instead of $v(P/R)$. Given a linear program P , a nonbasis \overline{N} of P , and $R \subseteq E$, then $e \in E$ is called *active with respect to* (N, R) if $v(R \cup \{e\}) \geq v(N)$. The set of all active constraints is denoted by $A(N, R)$, and its cardinality by $a_{N,R} := |A(N, R)|$. This definition has the following relation to the corresponding definition of enforcing constraints from the original MSW paper [10]: e enforcing in (R, B) if and only if $e \notin A(N, R)$. The following two *monotonicity relations* hold: $S \supseteq R \Rightarrow A(N, S) \subseteq A(N, R)$, and $v(\tilde{N}) \geq v(N) \Rightarrow A(\tilde{N}, R) \subseteq A(N, R)$. For $R \subseteq N$ we have $N \subseteq A(N, R)$ and hence $n \leq a_{N,R}$.

Definition 5 ($E(D, R)$ and $\beta(k, \ell)$) A call $\mathbf{MSW}(D, R)$ will generate a (finite or infinite) sequence of pivot operations, where a certain sequence is chosen according to the probability distribution implicitly defined by the algorithm. In the same way the length of such a sequence is probabilistic. We denote by $E(D, R)$ the *expected number of pivot operations* generated by $\mathbf{MSW}(D, R)$. For $k \geq 0$ and $\ell \geq 0$ let $\beta(k, \ell)$ denote the *smallest upper bound for $E(D, R)$* for all feasible dictionaries D that define a nondegenerate linear program and all $R \subseteq N$ such that $a_{N,R} \leq n + k$ and $|N \setminus R| = \ell$.

Lemma 6 *The function β holds:*

(i) *For any $k \geq 0$: $\beta(k, 0) = 0$.*

(ii) *For any $\ell \geq 0$: $\beta(0, \ell) = 0$.*

(iii) *For any $k \geq 1, \ell \geq 1$: $\beta(k, \ell) \leq \beta(k, \ell - 1) + \frac{1}{\ell} \sum_{j=1}^{\min(k, \ell)} (1 + \beta(k - j, \ell))$.*

Proof (Outline):

(i) $\ell = 0$ implies $R = N$, so there will be no pivot: $E(D, R) = 0$ and hence $\beta(k, 0) = 0$.

(ii) $D = D(\bar{B})$ is terminal for P/R , since otherwise $n < a_{N,R}$ in contradiction to $k = 0$.

(iii) We order the elements in $N \setminus R = \{e_1, \dots, e_\ell\}$ such that $v(R \cup \{e_1\}) \leq \dots \leq v(R \cup \{e_\ell\})$. We choose with probability $\frac{1}{\ell}$ an element $e = e_j \in N \cap R$. Then we compute recursively $\tilde{D} = \mathbf{MSW}(D, R \cup \{e\})$ with an expected number of $E(D, R \cup \{e\})$ pivot operations which is at most $\beta(k, \ell - 1)$. If \tilde{D} is terminal there are no more pivot operations, otherwise we make one simplex pivot to \hat{D} and recursively compute $\mathbf{MSW}(\hat{D}, R)$ which causes together the expected number of $1 + E(\hat{D}, R)$ pivot operations. We show that $E(\hat{D}, R) \leq \beta(k - j, \ell)$: Monotonicity of the simplex pivots implies $v(N) < v(\hat{N})$ and hence $A(\hat{N}, R) \subseteq A(N, R)$; for $i \leq j$ is $e_i \in A(N, R) \setminus A(\hat{N}, R)$; i.e. $a_{\hat{N}, R} \leq a_{N, R} - j \leq n + k - j$. $n \leq a_{\hat{N}, R}$ implies $j \leq k$, so we have to extend the sum of all possible cases not further than $\min(k, \ell)$.

□

Theorem 7 *For any $k \geq 0, \ell \geq 0$: $\beta(k, \ell) \leq e^{4 \cdot \sqrt{k \ln(\ell+1)}}$.*

Proof: The inequality holds for any function β with the properties as in Lemma 6. For a proof see [10], Proposition 5 (the function β_{MSW} there is $\beta + 1$). □

Proof of Theorem 3: Remark that $a_{N, \emptyset} \leq m + n$, hence the bound $\beta(m, n)$ is valid for any linear programs with $m = |B|, n = |N|$ for a call of $\mathbf{MSW}(D, \emptyset)$. We prove the finiteness and correctness according to the recursion of the algorithm and use induction in $k + \ell$. Let $B(k, \ell)$ denote the maximal number of pivot operations for any linear program with k and ℓ as defined for $\beta(k, \ell)$. As in the proof of Lemma 6 we can show: For any $k \geq 0$: $B(k, 0) = 0$; for any $\ell \geq 0$: $B(0, \ell) = 0$; for any $k \geq 1, \ell \geq 1$: $B(k, \ell) \leq B(k, \ell - 1) + B(k - 1, \ell) + 1$. We easily prove $B(k, \ell) \leq 2^{k+\ell} - 1$. The correctness of the algorithm is now trivial. □

In 1992 Kalai [6] presented three algorithms S_0, S_1, S_2 ; Goldwasser [5] showed in 1995 that a certain variant of S_0 is dual to **MSW**. In an article from 1997 Kalai [7] describes two new variants **Algorithm I** and **Algorithm II** and claims (without proof) that **Algorithm I** “is equivalent (in a dual-setting) to the Sharir-Welzl algorithm” of [14] which is exactly the same as the **MSW** algorithm of [10]. Kalai’s **Algorithm I** is the origin of the algorithm **Kalai** which is investigated in this subsection. We will compare **Kalai** with **MSW** in Subsection 4.3 with the result that the two algorithms are slightly different.

Kalai’s **Algorithm I** is not a complete algorithm. One difficulty is that Kalai describes some algorithmic steps without giving the order of their execution in the algorithm. Another weakness of the description is the missing definition of the trivial case of the recursion. Above all, there is no pivot operation explicitly declared. We tried to solve these problems in a most natural way. This leads to the form of **Kalai**. We remark that also for some other forms of Kalai’s algorithm the results of Subsection 4.3 hold.

We write $v(R)$ instead of $v(P/R)$. Given a linear program P , a nonbasis \bar{N} of P , and $R \subseteq E$, then $e \in E$ is called *strongly active with respect to* (N, R) if $v(R \cup \{e\}) > v(N)$ and $e \notin R$. The set of all strongly active constraints is denoted by $\tilde{A}(N, R)$ and its cardinality by $\tilde{a}_{N,R} := |\tilde{A}(N, R)|$. We remark that the above definition is the same as the definition of an active constraint in Kalai’s paper [7]. The *monotonicity relations* $S \supseteq R \Rightarrow \tilde{A}(N, S) \subseteq \tilde{A}(N, R)$ and $v(\tilde{N}) \geq v(N) \Rightarrow \tilde{A}(\tilde{N}, R) \subseteq \tilde{A}(N, R)$ hold.

Definition 8 ($\tilde{E}(D, R)$ and $\tilde{\beta}(k, \ell)$) We denote by $\tilde{E}(D, R)$ the *expected number of pivot operations* generated by **Kalai**(D, R) (analogous to $E(D, R)$ of Definition 5). For $k \geq 0$ and $\ell \geq 1$ let $\tilde{\beta}(k, \ell)$ denote the *smallest upper bound for* $\tilde{E}(D, R)$ for all feasible dictionaries D that define a nondegenerate linear program and all $R \subseteq N$ such that $\tilde{a}_{N,R} \leq k$ and $|N \setminus R| = \ell$.

Lemma 9 *The function $\tilde{\beta}$ holds:*

- (i) For any $k \geq 1$: $\tilde{\beta}(k, 1) \leq 1$.
- (ii) For any $\ell \geq 1$: $\tilde{\beta}(0, \ell) = 0$.
- (iii) For any $k \geq 1, \ell \geq 2$: $\tilde{\beta}(k, \ell) \leq \tilde{\beta}(k-1, \ell-1) + \frac{1}{\ell-1} \sum_{j=1}^{\min(k, \ell-1)} \tilde{\beta}(k-j, \ell)$.

Proof (Outline):

- (i) Either D is terminal and then there is no pivot operation, or we have a problem of dimension 1 which will be solved with one simplex pivot: $\tilde{E}(D, R) \leq 1$ and hence $\tilde{\beta}(k, 1) \leq 1$.
- (ii) $k = 0$ implies $\tilde{a}_{N,R} = 0$, and then $D = D(B)$ is terminal for R (since otherwise $0 < \tilde{a}_{N,R}$).
- (iii) We order the elements in $N \setminus R = \{e_1, \dots, e_\ell\}$ such that $v(R \cup \{e_1\}) \leq \dots \leq v(R \cup \{e_\ell\})$. If D is terminal for R then there is no pivot operation. Since $\ell \geq 2$ the case $|N \setminus R| = 1$

is not possible. If D is not terminal for R then we have to distinguish two cases: when the dictionary D is *nearly optimal* with redundant constraint $\tilde{e} \in N$, then $\tilde{e} = e_1$ and we choose with probability $\frac{1}{\ell-1}$ an element $e = e_j$ for some $j \in \{2, \dots, \ell\}$; recursion leads (similarly to the analysis of **MSW**) to the inequality $\tilde{\beta}(k, \ell) \leq \tilde{\beta}(k-1, \ell-1) + \frac{1}{\ell-1} \sum_{j=1}^{\min(k, \ell-1)} \tilde{\beta}(k-j, \ell)$. In the case that D is *not nearly optimal* we choose with probability $\frac{1}{\ell}$ an element $e = e_j$ for some $j \in \{1, \dots, \ell\}$, and we will obtain the inequality $\tilde{\beta}(k, \ell) \leq \tilde{\beta}(k-1, \ell-1) + \frac{1}{\ell} \sum_{j=1}^{\min(k, \ell)} \tilde{\beta}(k-j, \ell)$. Since according to the definition of $\tilde{\beta}$ for $i \leq j$ holds $\tilde{\beta}(k-j, \ell) \leq \tilde{\beta}(k-i, \ell)$, the right hand side of the first recursion inequality dominates the right hand side of the second. \square

Theorem 10 For any $k \geq 0, \ell \geq 1$: $\tilde{\beta}(k, \ell) \leq e^{4\sqrt{k \ln \ell}} + 1$.

Proof: Kalai [7] does not give a proof but the reference to MSW [10]. In fact we can use Theorem 7 as follows: Define $\hat{\beta}(k, \ell) := \tilde{\beta}(k, \ell+1) - 1$, and let $\bar{\beta}$ be the function which holds Lemma 6 everywhere with equality. Then Lemma 9 implies $\hat{\beta}(k, \ell) \leq \bar{\beta}(k, \ell)$, so by Theorem 7 for any $k \geq 0, \ell \geq 0$: $\hat{\beta}(k, \ell) \leq e^{4\sqrt{k \ln(\ell+1)}}$. This is equivalent to the claim. \square

Proof of Theorem 4: Remark that $\tilde{a}_{N, \emptyset} \leq m + n$, hence the bound $\beta(m+n, n)$ is valid for any linear programs with $m = |B|, n = |N|$ for a call of **Kalai**(D, \emptyset). We prove the finiteness and correctness according to the recursion of the algorithm and use induction in $k + \ell$. Let $\tilde{B}(k, \ell)$ denote the maximal number of pivot operations for any linear program with k and ℓ as defined for $\tilde{\beta}(k, \ell)$. As in the proof of Lemma 9 we can show: For any $k \geq 1$: $\tilde{B}(k, 1) \leq 1$; for any $\ell \geq 1$: $\tilde{B}(0, \ell) = 0$; for any $k \geq 1, \ell \geq 2$: $\tilde{B}(k, \ell) \leq \tilde{B}(k-1, \ell-1) + \tilde{B}(k-1, \ell)$. We easily prove $\tilde{B}(k, \ell) \leq \binom{k-1}{\ell-1}$ for $k \geq 1, \ell \geq 1$. The correctness of the algorithm is now trivial. \square

4.3 Comparison of Algorithms

We give a definition for the equivalence of algorithms and two examples for the comparison of the two algorithms **MSW** and **Kalai** that have been presented in the previous two subsections. We show that these algorithms are not equivalent in our sense.

Definition 11 (Probabilistic Pivot Sequence Distribution $\mathcal{P}(D, R)$) Consider a randomized pivoting algorithm **A**(D, R) accepting as input a dictionary D and a set of indices R (e.g. as in **MSW**). After the call of **A**(D, R) the algorithm will generate a (*probabilistic*) *pivot sequence*, i.e. subject to the randomization of the algorithm a sequence of pivot operations on pivots $(i_1, j_1), (i_2, j_2), \dots$. If the algorithm **A** with input D and R always terminates, all possible pivot sequences are finite and will occur with a certain probability: the corresponding probability distribution defined on the set of all finite pivot sequences for a call of **A**(D, R) is denoted by $\mathcal{P}(D, R)$.

Definition 12 (Equivalence of Algorithms) Two randomized pivoting algorithms **A1**(D, R) and **A2**(D, R), both accepting the same input (D, R) where D is a dictionary and R some set of indices, are called *equivalent*, if they have for every accepted input (D, R) the same probabilistic pivot sequence distribution $\mathcal{P}(D, R)$.

Example 1. The first example considered is given in dictionary form (1) by the dictionary

0	-1	3	-1
1	2	-1	-1
2	1	-1	0
2	-1	0	0

and $N = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$. The probabilistic pivot sequence distributions of **MSW** and **Kalai** are as follows:

Pivot Sequence	MSW	Kalai
$(4, 2), (5, 1), (6, 4)$	Prob. = $\frac{2}{3}$	Prob. = $\frac{3}{4}$
$(4, 2), (5, 1), (6, 3), (3, 4)$	Prob. = $\frac{1}{3}$	Prob. = $\frac{1}{4}$

Hence the two algorithms are not equivalent. The expected number of pivot operations is $3\frac{1}{3}$ for **MSW** and $3\frac{1}{4}$ for **Kalai**: For Example 1 **Kalai** is (slightly) “faster” than **MSW**. This is not true in general, as the following example shows:

Example 2. The second linear program is very similar to the first, we have to modify not more than one entry in the initial dictionary which is for Example 2

0	-1	3	-1
1	2	-1	-1
2	1	-1	$\frac{1}{2}$
2	-1	0	0

The probabilistic pivot sequence distributions of **MSW** and **Kalai** for this second example are

Pivot Sequence	MSW	Kalai
$(4, 2), (5, 1), (6, 3)$	Prob. = $\frac{1}{3}$	Prob. = $\frac{1}{4}$
$(4, 2), (5, 1), (6, 4), (4, 3)$	Prob. = $\frac{2}{3}$	Prob. = $\frac{3}{4}$

The expected number of pivot operations is $3\frac{2}{3}$ for **MSW** and $3\frac{3}{4}$ for **Kalai**.

5. Final Remarks

We have presented in one framework two randomized pivoting algorithms and compared them in order to show that they are not the same. The original discussions used very different settings which made comparison difficult. Our precise analyses may help us investigate where we possibly overestimate the real complexity. One goal is an extension of the framework to analyse a more general class of algorithms including primal-dual algorithms such as criss-cross methods [4].

Our framework is combinatorial in the sense that it uses only the signed incidence relations of the input data and not quantitative information: We can interpret the problems and algorithms as if they would be in a setting of oriented matroids instead of linear programming (see e.g. [1]). We even can deal with assumptions as (initial) feasibility and nondegeneracy of LP problems in the same combinatorial way. The techniques we may use here are e.g. symbolic bounding techniques

and combinatorial perturbation. We can translate the complexity results of Subsection 5.2 to related results for the overall complexity when this techniques are used; for a detailed discussion see e.g. [3]. All this makes us hope that we can develop a unifying framework for a rich class of (randomized) pivoting algorithms which then will be an excellent basis for further investigation of most challenging questions on the complexity of linear programming.

Acknowledgement. The authors wish to thank Günter Rote for fruitful discussions on this work.

Corresponding Author: L. Finschi
Institut für Operations Research, ETH Zürich
ETH Zentrum
8092 Zürich, Switzerland

References

- [1] R. G. Bland, A combinatorial abstraction of linear programming, *Journal of Combinatorial Theory*, series B 23, 1977, pp. 33–57.
- [2] G. B. Dantzig: *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.
- [3] L. Finschi: *Randomized Pivot Algorithms in Linear Programming*, Diploma Thesis, Institut für Operations Research, ETH Zürich, 1997.
- [4] K. Fukuda, T. Terlaky: Criss-cross methods: A fresh view on pivot algorithms, *Mathematical Programming*, 79, Elsevier North-Holland, 1997, pp. 369–395.
- [5] M. Goldwasser: A survey of linear programming in randomized subexponential time, *SIGACT News*, 26 no. 2, 1995, pp. 96–104.
- [6] G. Kalai, A subexponential randomized simplex algorithm, In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, 1992, pp. 475–482.
- [7] G. Kalai, Linear programming, the simplex algorithm and simple polytopes, *Mathematical Programming*, 79, Elsevier North-Holland, 1997, pp. 217–233.
- [8] N. Karmarkar: A new polynomial-time algorithm for linear programming, *Combinatorica*, 4, 1984, pp. 373–395.
- [9] L. G. Khachiyan: Polynomial algorithms in linear programming, *U.S.S.R Comput. Maths. Math. Phys.*, Vol. 20, No. 1, 1980, pp. 53–72.
- [10] J. Matoušek, M. Sharir, E. Welzl: A subexponential bound for linear programming, *Proceedings of the eighth annual symposium on Computational Geometry*, ACM, Berlin, 6/1992, pp. 1–8.
- [11] N. Megiddo: Linear programming in linear time when the dimension is fixed, *Journal of the Association for Computing Machinery*, 31, 1984, pp. 114–127.
- [12] R. Motwani, P. Raghavan: *Randomized Algorithms*, Cambridge University Press, 1995.
- [13] R. Seidel: Small-dimensional linear programming and convex hulls made easy, *Discrete & Computational Geometry* 6, Springer-Verlag, New York, 1991, pp. 423–434.
- [14] M. Sharir, E. Welzl: A combinatorial bound for linear programming and related problems, *STACS 92*, volume 577 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992, pp. 569–579.