Randomized Pivot Algorithms in Linear Programming

Lukas Finschi
Eidgenössische Technische Hochschule Zürich
Institut für Operations Research
Clausiusstrasse 45/47
CH-8092 Zürich
finschi@ifor.math.ethz.ch

August 22, 1997

Abstract

We discuss randomized algorithms for linear programming, in particular an algorithm due to Matoušek, Sharir, and Welzl with subexponential expected run time. This recursive algorithm (denoted by MSW) was used for linear programming under certain strong assumptions, so all LP problems to be recursively solved are bounded and have a unique optimal solution. Consequently it is not clear how one can transform the MSW algorithm to a true LP algorithm without losing its subexponential behaviour and without introducing large numbers (as e.g. when a numerical bounding box is used). In this paper we present a simple pivot algorithm based on MSW using a new notion of lexicographic bounding box. It is a sort of the dual simplex method with the same time complexity as MSW. The pivoting algorithm has been implemented and we report some computational experiences.

1 Introduction

A linear programming problem is to find a maximizer (or minimizer) of a linear function over a system of linear inequalities:

$$\max c^T x \text{ subject to } Ax \leq b, \tag{LP}$$

where A is given $n \times d$ matrix, c and b are given rational d- and n-vectors, and x is a variable d-vector. It is unnecessary to stress the theoretical and practical importance of linear programming (LP) since it is repeatedly witnessed in abundance of books, papers and commercial applications. The present paper is a modest attempt to improve our understanding of existing algorithms for LP problems. In particular our main concern is the number of elementary arithmetic operations necessary to solve LP problems. For this reason we assume each operation can be performed in constant time.

Dantzig [2] introduced in 1947 the simplex method which is very efficient in practice but requires in the worst case a number of pivot steps that is exponential in d. The first polynomial algorithms for linear programming were presented by Kachiyan [6] and Karmarkar [5]. The number of elementary arithmetic operations of these algorithms depends not only on d and n but also on the size of the coefficients c_i, A_{ij}, b_i. To find a strongly polynomial algorithm (for which the number of arithmetic operations depends only on d and n) remains to be one of the most challenging open questions.

In 1983 Megiddo [9] presented a (deterministic) algorithm which solves LP in linear time $O(n)$ when d is regarded to be fixed; the dependence on d was improved in the following but remained exponential. In 1991 Seidel [11] introduced a simple randomized incremental algorithm

with expected running time of $O(d!n)$; Sharir and Welzl presented in 1992 an improvement of Seidel's algorithm within an axiomatic framework, and after Kalai [4] proved (also in 1992) for the first time a subexponential expected running time for linear programming (in terms of d and n), Matoušek, Sharir and Welzl proved for the Sharir-Welzl algorithm (abbreviated by MSW) a subexponential bound of $O(dn \cdot e^{4\sqrt{d \ln(n-d+1)}})$ which is similar to Kalai's bounds; together with a randomized sampling technique of Clarkson from 1988 they achieved the currently best bound of $O(d^2 n + \ln n \cdot e^{O(\sqrt{d \ln d})})$.

The main motivation is to understand the MSW algorithm as a precise pivot algorithm so that we can compare it with the simplex method and any other pivot algorithm on the same ground. Goldwasser [3] showed that MSW and one of Kalai's algorithms are dual to each other. Our work is to extend his transformation so that the MSW algorithm (and hopefully any other geometric algorithm for LP) becomes precise and complete. Also this will help us implement the algorithm and investigate where we possibly overestimate the theoretical complexity.

There are several difficulties to accomplish this goal, including the relaxation of assumptions such as boundedness, nondegeneracy, and uniqueness of optima. The key idea used here are a weaker version of the LP-type framework of Sharir-Welzl and a new notion of combinatorial bounding box. While our new description remains to be quite simple, it can handle all general linear programs and its implementation is straightforward. Furthermore the complexity of the original algorithm carries over to this precise and complete algorithm.

2 Optimization Problems of LP-Type

We introduce the abstract framework of Sharir and Welzl [12] in a slightly generalized way, namely with additional the elements $+\infty$ *(unbounded or undefined problem)* and $-\infty$ *(infeasible problem)*:

Definition 1 (Optimization Problems of LP-Type) *Consider a pair (H, v) of a finite* constraint set *$H \neq \emptyset$ and of a* value function *$v : \mathcal{P}(H) \to \mathcal{L}$, where $\mathcal{P}(H)$ denotes the set of all subsets of H and $(\mathcal{L}, \leq)$ is a linearly ordered set with maximum $+\infty$ and minimum $-\infty$, i.e. for all $v \in \mathcal{L}$: $-\infty \leq v \leq +\infty$. A subset of constraints $B \subseteq H$ is called* a basis *if $v(B) < +\infty$ and for all $\tilde{B} \subsetneq B : v(\tilde{B}) > v(B)$. For any $G \subseteq H$: A set of constraints $B \subseteq H$ is called* an optimal basis of G *if B is a basis with $B \subseteq G$ and $v(B) = v(G)$. The following two properties will be used as axioms:*

- **Monotonicity:** *For all F, G with $F \subseteq G \subseteq H : v(F) \geq v(G)$*
- **Locality:** *For all F, G with $F \subseteq G \subseteq H$ and $v(F) = v(G) < +\infty$ and for any $h \in H$: $v(G) > v(G \cup \{h\}) \Rightarrow v(F) > v(F \cup \{h\})$*

If for (H, v) the axioms of monotonicity and locality hold, we say that (H, v) specifies an optimization problem of LP-type *(the problem is: find an optimal basis of H).*

If (H, v) specifies an optimization problem of LP-type, then also (G, v) for any $G \subseteq H$ (where the value function $v : \mathcal{P}(G) \to \mathcal{L}$ is defined by restriction on $\mathcal{P}(G)$), and there exists an optimal basis B of G if and only if $v(G) < +\infty$. We call $h \in H$ *violated by* $G \subseteq H$ if $v(G) > v(G \cup \{h\})$, and h *is extreme in* G if $v(G \setminus \{h\}) > v(G)$. We denote the cardinality of a set M by $|M|$.

Definition 2 (Combinatorial Dimension $\dim(H, v)$ of (H, v)) *Given an optimization problem of LP-type specified by (H, v). We define the combinatorial dimension of (H, v) by*

$$\dim(H, v) := \max\left(\{|B| \mid B \subseteq H \text{ a basis with } v(B) > -\infty\} \cup \{-1\}\right).$$

The combinatorial dimension will take the place of the usual dimension d of linear programs when we apply this framework to linear programming. The fundamental property is the following: If $v(G) \notin \{-\infty, +\infty\}$ for $G \subseteq H$, then G has at most $\dim(H, v)$ extreme constraints.

3 The Algorithm of Matoušek, Sharir, Welzl

We consider (H, v) specifying an optimization problem of LP-type; assume that we have two computational capabilities: We are able to test violations of the form "given any basis $\tilde{F} \subseteq H$ and any constraint $h \in H$, is h violated by $\tilde{F}$?", and we have an algorithm **Basis** which computes for a basis $\tilde{F} \subseteq H$ and any constraint $h \in H$ which is violated by $\tilde{F}$ an optimal basis of $\tilde{F} \cup \{h\}$. The algorithm of Matoušek, Sharir, Welzl is then as follows:

Algorithm MSW:
Input: $G \subseteq H$, and a basis $F \subseteq G$.
Output: An optimal basis B of G.

begin MSW(G, F);
  if $G = F$ **then return** F
  else
    choose $h \in G \setminus F$ at random (all $h \in G \setminus F$ have the same probability $\frac{1}{|G \setminus F|}$);
    $\tilde{F} :=$ **MSW**$(G \setminus \{h\}, F)$; *(first type recursive call)*
    if h not violated by $\tilde{F}$ **then return** $\tilde{F}$
    else
      $\hat{F} :=$**Basis**$(\tilde{F} \cup \{h\})$;
      if $v(\hat{F}) = -\infty$ **then return** $\hat{F}$
      else return MSW$(G, \hat{F})$ *(second type recursive call)*
      endif
    endif
  endif
end MSW.

The algorithm has two recursive calls, the first is mandatory (unless we are in the trivial case $G = F$) and solves a problem with one constraint h removed, the second recursive call only occurs when the removed constraint is violated by the solution of the relaxed problem. The efficiency of the algorithm will arise from the fact that constraint violations and by this second type recursive calls are rather rare, and if we have to call **MSW**$(G, \hat{F})$, then $\hat{F}$ keeps information about the previous solution $\tilde{F}$ which will guarantee that the recursion has to stop at a certain level and allows to prove the subexponential upper bound for the expected running time.

So the power of **MSW**(G, F) is hidden in the second input argument F, but remark that a basis F has to be given before **MSW** is started, and such a basis exists if and only if $v(G) < +\infty$; even if $v(G) < +\infty$, it might be not easy to find a basis F.

The next theorem gives the analysis of the algorithm **MSW** (see [7]):

Theorem 3 (Analysis of MSW) *For any F, G with $F \subseteq G \subseteq H$, F a basis:*

(i) **MSW***(G, F) terminates after at most $2^{|G|+1} - 2$ further calls of* **MSW***, and the output is an optimal basis of G.*

(ii) If $v(F) > -\infty$ and if (H, v) is basis regular (i.e.: for every basis $\bar{F}$ with $v(\bar{F}) > -\infty$ holds $|\bar{F}| = d$), then the expected running time of **MSW***(G, F) is at most*

$$e^{4 \cdot \sqrt{d \ln(|G|-d+1)}} \cdot (T_{\textbf{Basis } \textit{Computation}} + O(|G|) \cdot T_{\textit{Violation Test}} + O(|G|)),$$

where $d := \dim(H, v)$, and $T_{\textbf{Basis } \textit{Computation}}$ and $T_{\textit{Violation Test}}$ denote the costs for the indicated computations.

When we apply the abstract framework in the context of linear programming problems, the LP-type problems will be basis regular.

Remark that the violation test and the computation of an optimal basis for a constraint set of the form $\tilde{F} \cup \{h\}$ have to be problems where the solution can be obained with low computational costs: The analysis of **MSW** does rely on these computational primitives.

There exist examples of LP-type problems for which the algorithm **MSW** has an expected run time close to the upper bound (see [8]); however, there are no known linear programming problems with such a long running time.

4 Transformations from LP to LP-Type

Given a linear programming problem (LP), we have to define an optimization problem of LP-type (H, v), i.e. we have to define H and for every $G \subseteq H$ its value $v(G)$ such that monotonicity and locality hold. A natural way to do this would be: Let be H the set of linear constraints given by $Ax \leq b$ and for $G \subseteq H$ consider the relaxed linear program

$$\max c^T x \text{ subject to } A_G x \leq b_G, \qquad ((LP)\mid_G)$$

where the constraint set given by $A_G x \leq b_G$ corresponds to G. Then define $v(G) := c^T x^G$, where $x^G \in \mathbb{R}^d$ is an optimal solution of $(LP)\mid_G$; if the linear program is infeasible, set $v(G) := -\infty$, if the linear program is unbounded (with respect to maximization), set $v(G) := +\infty$. This attempt fails: The locality condition might be violated (namely when the optimal solution x^G is not unique).

We solve this *first transformation problem*, namely the *uniqueness of the optimal solution*, as follows: Instead of $(LP)\mid_G$ we solve

$$\max(c^T x, x_1, \ldots, x_d) \text{ subject to } A_G x \leq b_G, \qquad ((LP)\mid_G^{lex})$$

where maximization is done with respect to the lexicographical order given by

$$(a_1, \ldots, a_{k_a}) \leq (b_1, \ldots, b_{k_b}) :\Leftrightarrow \begin{cases} \text{Either } k_a \leq k_b \text{ and } \forall i \in \{1, \ldots, k_a\}\colon a_i = b_i, \\ \text{or } \exists j \in \{1, \ldots, \min\{k_a, k_b\}\}\colon \forall i < j\colon a_i = b_i, \text{ and } a_j < b_j. \end{cases}$$

We set the value $v(G)$ for infeasible or unbounded problems $(LP)\mid_G^{lex}$ as before, but now for bounded and feasible problems $v(G) := (c^T x^G, x_1^G, \ldots, x_d^G)$, where x^G is the unique optimal solution of $(LP)\mid_G^{lex}$; boundedness of $(LP)\mid_G^{lex}$ is defined with respect to the lexicographical maximization. In fact we have defined an appropriate transformation:

Theorem 4 *Given a problem LP, then let H be the set of constraints in $Ax \leq b$, and for $G \subseteq H$ we consider the corresponding problem $(LP)\mid_G^{lex}$; when we set*

$$v(G) := \begin{cases} -\infty & \text{if } (LP)\mid_G^{lex} \text{ is infeasible,} \\ (c^T x^G, x_1^G, \ldots, x_d^G) & \text{if } (LP)\mid_G^{lex} \text{ is feasible and bounded and has optimal solution } x^G, \\ +\infty & \text{if } (LP)\mid_G^{lex} \text{ is unbounded,} \end{cases}$$

then (H, v) specifies an optimization problem of LP-type with $\dim(H, v) \leq d$; (H, v) is basis regular.

Proof: We have to show monotonicity, locality, and that every basis B with $v(B) > -\infty$ has cardinality d.

- **Monotonicity:** Consider F, G with $F \subseteq G \subseteq H$: If $v(F) = -\infty$ (i.e. $(LP)\,|_F^{lex}$ is infeasible and hence also $(LP)\,|_G^{lex}$) or if $v(F) = +\infty$, the inequality $v(F) \geq v(G)$ obviously holds. If $v(F) = (c^T x^F, x_1^F, \ldots, x_d^F)$ (i.e. $(LP)\,|_F^{lex}$ is bounded and has optimal solution x^F), then $(LP)\,|_G^{lex}$ is either infeasible (hence $v(G) = -\infty < v(F)$), or has a finite optimum x^G; since x^G is feasible for $(LP)\,|_F^{lex}$, the optimality of x^F implies $v(G) = (c^T x^G, x_1^G, \ldots, x_d^G) \leq (c^T x^F, x_1^F, \ldots, x_d^F) = v(F)$.

- **Locality:** Given F, G with $F \subseteq G \subseteq H$, $v(F) = v(G) < +\infty$, and a constraint $h \in H$ such that $v(G) > v(G \cup \{h\})$: $v(F) = -\infty$ is not possible: Monotonicity implies $-\infty = v(F) \geq v(G) > v(G \cup \{h\}) \geq -\infty$, a contradiction; so $v(F) = (c^T x^F, x_1^F, \ldots, x_d^F)$. Then (with $v(F) = v(G)$) x^F is the optimal solution of $(LP)\,|_G^{lex}$, too. $v(G) > v(G \cup \{h\})$ implies that x^F is not feasible for the constraint h, so $v(F) \neq v(F \cup \{h\})$, i.e. (with monotonicity) $v(F) > v(F \cup \{h\})$.

- **Cardinality of a basis:** Let be B a basis with $v(B) > -\infty$ (if such a basis does not exist, then $-1 = \dim(H, v) < d$): The definition of a basis implies $v(B) < +\infty$ and $v(B) = (c^T x^B, x_1^B, \ldots, x_d^B)$, then the optimal solution x^B of $(LP)\,|_B^{lex}$ is a vertex of the feasible region of $(LP)\,|_B^{lex}$; this vertex can be defined as the optimal solution of $(LP)\,|_{\tilde{B}}^{lex}$ for $\tilde{B} \subseteq B$ with $|\tilde{B}| = d$; the definition of a basis implies $B = \tilde{B}$ and hence $|B| = d$.

■

When we consider solvability of such LP-type optimization problems and the applicability of algorithms as **MSW**, then we see that there is a *second transformation problem*, namely the *existence and detection of a basis* (as we need it as second input argument F for **MSW**). If (LP) is unbounded, there does not exist any basis; if (LP) is bounded, we still have to find a basis F. We solve this second transformation problem by bounding the problem (LP), namely we add a constraint set X which has the property that for $G \subseteq H$ the optimal solution of $(LP)\,|_{G \cup X}$ is on a box constraint only if $(LP)\,|_G$ is unbounded (and the boxed problem has to be infeasible only if the unboxed is infeasible).

We consider the box constraints X as a part of H, so the number of constraints increases by $|X|$; (H, v) is basis regular and all problems which are solved by **Basis** have exactly $d + 1$ constraints (if we would use X as a fix bounding box, i.e. H is the same set as without bounding box but for every $G \subseteq H$ we solve the problem with constraints $G \cup X$, then both properties, the basis regularity and the mentioned input cardinality of **Basis** would not hold). For **MSW**, a starting basis F can be obtained easily as a subset of X defining a bounding corner of the box.

The addition of *numerical* bounding box contraints is affected with a number of drawbacks: The computation of the size of the box can be difficult, and in practice we overestimate the size by far so we have to deal with very large numbers; furthermore we can not design a purely combinatorial algorithm with a numerical bounding box as additional constraint set. In order to avoid such problems we use the concept of a bounding box in a *combinatorial* manner. Inspired by Seidel [11] (there some square box is used for the description of an algorithm) we developped a new bounding technique for linear programming which is also useful for the transformation of LP to other axiomatics than LP-type or in other contexts than linear programming (as e.g. for oriented matroids).

Without loss of generality we assume that the linear programming problem is of the form

$$\max c^T x \text{ subject to } Ax \leq b,\ x \geq 0,\ x \in \mathbb{R}^d, \qquad ((LP)_{\geq 0})$$

where $A \in \mathbb{R}^{(n-d) \times d}$, $b \in \mathbb{R}^{n-d}$, $c \in \mathbb{R}^d$. Now we introduce a parameter L and add box constraints of the form $x_i \leq L^i$ (the i in L^i is an exponent and not only an index), so with lexicographical optimization as above we obtain

$$\max(c^T x, x_1, \ldots, x_d) \text{ subject to } Ax \leq b, \text{ for all } i \in \{1, \ldots, d\} : 0 \leq x_i \leq L^i. \qquad ((LP)\,|^L)$$

We call the box $0 \le x_i \le L^i$ a *lexicographical bounding box* or in short terms an *L-box*. If L is set to a real number, $(LP)\ |^L$ is a linear program with constraint set H, $|H| = n + d$; for a $G \subseteq H$ we solve as above the corresponding problem $(LP)\ |^L_G$ and define $v(G)$ as above (with $x^G = x^G(L)$ depending on L). For large values of L we have the desired properties of a bounding box. The important thing is that we don't have to know how large L must be, i.e. we can compute with the L-box just combinatorially as if L would be an arbitrary large number; in addition the lexicographical bounding box has a shape which leads to preferable properties when compared with a numerical bounding box.

We add some considerations which are not necessary when just working with the L-box technique for **MSW** but which offer some insight in its combinatorial structure. By the following theorem the optimal solution $x^G(L)$ of problems $(LP)\ |^L_G$ for large L can be expressed in terms of vectors $u_G, w^1_G, \ldots, w^d_G \in \mathbb{R}^d$ which are independent of L; this step introduces a combinatorial notion of the optimal solution of unbounded linear programs.

Theorem 5 *There exists $L_0 \ge 0$ such that for all $L \ge L_0$: Either the problem $(LP)\ |^L_G$ is infeasible or unbounded, or there exist (unique) u_G, $w^1_G, \ldots, w^d_G$ which are independent from L such that the problem $(LP)\ |^L_G$ has the optimal solution $x^G = u_G + \sum_{i=1}^d L^i w^i_G$.*

We can give now another definition of the value $v(G)$ which is equivalent to the previous for large L and independent from L, namely we consider $(LP)\ |^L_G$ for $L \ge L_0$ and theorem 5: If $(LP)\ |^L_G$ is infeasible we set $v(G) := -\infty$, if it is unbounded $v(G) := +\infty$, otherwise (with $u := u_G$, $w^1 := w^1_G, \ldots, w^d := w^d_G$)

$$v(G) := (c^T w^d, \ldots, c^T w^1, c^T u,\ w^d_1, \ldots, w^1_1, u_1,\ w^d_2, \ldots, w^1_2, u_2, \ldots, w^d_d, \ldots, w^1_d, u_d).$$

5 The Algorithm of Matoušek, Sharir, Welzl in Pivot Form

We discuss in this section the pivot form of the algorithm of Matoušek, Sharir, Welzl which we call **MSWPivot** (for the original algorithm see **MSW** in section 3). For the linear programs we use the notation of dictionaries as presented in the next subsection. In the second subsection we describe the relation between the pivot form and the original setting of the algorithm. The third subsection contains a criteria for the optimality of a dictionary and discusses the problem of the pivot search, i.e. the equivalent of the computation of **Basis**$(\tilde{F} \cup \{h\})$ in **MSW**. In subsection 5.4 finally we present the algorithm and its analysis.

5.1 Dictionaries

We will use the following matrix notation: For two finite and nonempty sets I_R, I_C we consider a *matrix* $M \in \mathbb{R}^{I_R \times I_C}$ which consists of the *elements* (or *entries*) M_{ij} for $i \in I_R$, $j \in I_C$. For nonempty subsets $S_R \subseteq I_R$, $S_C \subseteq I_C$ we denote the submatrix corresponding to the entries M_{ij} for $i \in S_R$, $j \in S_C$ by $M_{S_R S_C}$. We use the following abbreviations: $M_{S_R} := M_{S_R I_C}$ and $M_{\cdot S_C} := M_{I_R S_C}$. For index sets with only one element we omit the braces: $M_i := M_{\{i\}}$ is the row of M with index $i \in I_R$, $M_{\cdot j} := M_{\cdot \{j\}}$ is the column of M with index $j \in I_C$.

In the next definition we introduce basis and dictionary; we think that there can hardly be confusion of this definition of a basis and the definition in the context of optimization problems of LP-Type:

Definition 6 (Basis and Nonbasis; Dictionary of a Basis) *Given a matrix $M \in \mathbb{R}^{I_R \times I_C}$ of rank $|I_R|$. A* basis (of M) *is a subset $B \subseteq I_C$ such that $|B| = |I_R|$ and $M_{\cdot B}$ is regular. A* nonbasis (of M) *is a subset $N \subseteq I_C$ such that $B = I_C \setminus N$ is a basis. When we use in the same context B and N, then B is a basis and $N = I_C \setminus B$.*
Given a basis $B \subseteq I_C$. Then we call $D \equiv D(B) := -(M_{\cdot B})^{-1} M_{\cdot N} \in \mathbb{R}^{B \times N}$ the dictionary of B.

As elementary property we have for any basis B of M, $D = D(B)$, and $\xi \in \mathbb{R}^{I_C}$ the relation $M\xi = 0 \Leftrightarrow \xi_B = D\xi_N$. The next definition is fundamental for the algorithms:

Definition 7 (Pivot Operation on (i,j)) *Given $M \in \mathbb{R}^{I_R \times I_C}$ of rank $|I_R|$, B a basis of M, $D = D(B)$, $i \in B$, $j \in N$ such that $D_{ij} \neq 0$: The replacement of B by $B \setminus \{i\} \cup \{j\}$ is called* the pivot operation on (i,j). *(i,j) is called* the pivot, *D_{ij} is called* the pivot element of the dictionary.

Lemma 8 *Given $M \in \mathbb{R}^{I_R \times I_C}$ of rank $|I_R|$, B a basis of M, $D = D(B)$, $i \in B$, $j \in N$ such that $D_{ij} \neq 0$: Then is $B \setminus \{i\} \cup \{j\}$ again a basis of M.*

If $D_{ij} = 0$, then it is not possible to pivot on (i,j). For $i \in B$ and $j \in N$ with $D_{ij} \neq 0$, we can compute the new dictionary $\tilde{D} = D(B \setminus \{i\} \cup \{j\})$ after a pivot operation on (i,j) by substitution of variable ξ_j; this leads to the rules of a (dual) pivot step, and $\tilde{D}$ can be computed from D with costs of $O(|B| \cdot |N|)$.

We transform $(LP)\ |^L$ to dictionary notation: We introduce new variables $y_1, \ldots, y_{n-d}$ (as slack variables for $Ax + y = b$) and a variable $z := c^T x$, and when we take the constant 1 into the variable set ξ with the correspondence

$$\xi_0 \equiv 1,\ \xi_1 \equiv x_1,\ \ldots,\ \xi_d \equiv x_d,\ \xi_{d+1} \equiv y_1,\ \ldots,\ \xi_n \equiv y_{n-d},\ \xi_{n+1} \equiv z,$$

we can write $(LP)\ |^L$ in the *the dictionary form* (with $X := \{1, \ldots, d\}$ and $Y := \{d+1, \ldots, n\}$):

$$\begin{array}{lll} \max(z, x_1, \ldots, x_d) & \text{subject to} & \xi_B = D\xi_N, \\ & & \text{for all } i \in X : 0 \leq \xi_i \leq L^i, \\ & & \text{for all } i \in Y : 0 \leq \xi_i, \end{array} \tag{1}$$

where D is the dictionary of a basis B of the matrix

$$M = \left(\begin{array}{c|c|c} -b & A & \\ \hline 0 & -c^T & \mathbb{1} \end{array} \right) \in \mathbb{R}^{\{d+1,\ldots,n+1\} \times \{0,\ldots,n+1\}}.$$

Given a dictionary D, we usually like to identify it with a certain point $x(D) \in \mathbb{R}^d$, namely the intersection of the (binding) nonbasis constraints. Since a variable $i \in X$ stands for two constraints, i.e. $0 \leq x_i$ and $x_i \leq L^i$, we will use for variables $i \in N \cap (X \cup Y)$ a flag which has either the value "i-bound = lowerbound" or "i-bound = upperbound". *From now we assume that a dictionary is always given with such flags.* Since we will always have $0 \in N$ and $n+1 \in B$, we can define:

Definition 9 *Given a dictionary $D = D(B)$, then we define the current value $\xi^=$ of ξ as follows: We set for $i \in N$*

$$\xi_i^= := \begin{cases} 1 & \text{if } i = 0, \\ 0 & \text{if } i \in X \cup Y \text{ with flag "i-bound = lowerbound"}, \\ L^i & \text{if } i \in X \cup Y \text{ with flag "i-bound = upperbound"}; \end{cases}$$

then we can define $\xi_B^= := D\xi_N^=$ and $x(D) := (x_1^=, \ldots, x_d^=)$.

5.2 Relation of the Pivoting Algorithm to the Axiomatic Framework

For the transformation of a linear program to the axiomatic framework of section 2 we use the techniques from section 4. The input of **MSW** consists of two sets of constraints G and F with $F \subseteq G \subseteq H$ and F a basis; the input of **MSWPivot** are a set $M \subseteq X \cup Y$ of socalled *marked* constraints and a dictionary D (including bound flags). The following definition defines a mapping from M and D to G and F:

Definition 10 ($G(M)$ and $F(D)$) *For a set $M \subseteq X \cup Y$ with $N \cap (X \cup Y) \subseteq M$ and a dictionary D we define two constraint sets $G(M)$ and $F(D)$:*

- *$G(M)$ consists of the constraints $0 \leq \xi_i \leq L^i$ for $i \in M \cap X$ and $0 \leq \xi_i$ for $i \in M \cap Y$,*
- *$F(D)$ consists of the constraints $0 \leq \xi_i$ for $i \in N \cap (X \cup Y)$ with bound flag "i-bound = lowerbound" and $\xi_i \leq L^i$ for $i \in N \cap (X \cup Y)$ with bound flag "i-bound = upperbound".*

$H = G(X \cup Y)$ is the set of all constraints, and because of $N \cap (X \cup Y) \subseteq M$ we have always $F \subseteq G \subseteq H$. The point $x(D)$ which is identified with the dictionary D (see definition 9) is exactly the intersection of the constraints in $F(D)$. Remark that with definition 10 the n variables in $X \cup Y$ represent $d+n$ constraints, since every $i \in X$ stands for two constraints. The following definition determines (for the situations where we need it) for a $i \in X \cup Y$ a unique constraint h_i:

Definition 11 ($i \in X \cup Y$ violated by D; h_i) *Given a dictionary D with basis B, where $z \in B$, and any variable $i \in X \cup Y$. The variable i is called* violated by D, *when either $\xi_i^= < 0$ (then we denote the constraint $\xi_i \geq 0$ by h_i), or $i \in X$ and $\xi_i^= > L^i$ for large L (then h_i denotes the constraint $\xi_i \leq L^i$); for $\xi_i^=$ see definition 9. If $i \in N$ (and hence not violated by D), then h_i denotes the ξ_i constraint in $F(D)$, i.e. $\xi_i \geq 0$ if "i-bound = lowerbound", $\xi_i \leq L^i$ otherwise.*

In **MSWPivot** the value $v(G)$ for some $G \subseteq H$ is represented by a pair (D, i) where D is a dictionary (with basis B and nonbasis N) and $i \in B$: The value $v(G) = (c^T x^G, x_1^G, \ldots, x_d^G)$ corresponds to $(D, n+1)$ with $x^G = x(D)$ (depending on L if and only if there is a flag "i-bound = upperbound"), the value $v(G) = -\infty$ is represented by (D, i) such that i is violated by D and the $d+1$ constraints in $F(D) \cup \{h_i\}$ define an infeasible linear program; $v(G) = +\infty$ will never occur.

5.3 Optimality Criteria for Dictionaries and Pivot Search in MSWPivot

In this subsection we study the following problem (the *problem of the pivot search*), which corresponds to the computation of **Basis**$(\tilde{F} \cup \{h\})$ in the algorithm **MSW**: For large L (i.e. $L \geq L_0$), a dictionary D (with basis B and nonbasis N) such that $x(D)$ is the optimal solution of $(LP)\,|^L_{F(D)}$, and a variable $i \in B \cap (X \cup Y)$ such that i is violated by D, determine whether $(LP)\,|^L_{F(D)\cup\{h_i\}}$ is infeasible or not; if not, then find the (unique) $j \in N \cap (X \cup Y)$ such that $x(\tilde{D})$ is optimal for $(LP)\,|^L_{F(D)\cup\{h_i\}}$, where $\tilde{D}$ denotes the dictionary after a pivot operation on (i, j).

We describe first a trivial (but unefficient) method which solves the pivot search: When we have a criteria which decides for any dictionary D whether $x(D)$ is optimal for $(LP)\,|^L_{F(D)}$, then we can just compute for every $j \in N \cap (X \cup Y)$ the dictionary $\tilde{D}$ after a pivot operation on (i, j): If j is not violated by $\tilde{D}$ and $\tilde{D}$ is optimal for $(LP)\,|^L_{F(\tilde{D})}$ (here we need the criteria), then j is the answer for the pivot search. The costs then are $O(d) \cdot (T_{\text{PivotOperation}} + T_{\text{Criteria}})$. In fact we have such an optimality criteria:

Theorem 12 (Optimality Criteria for a Dictionary D) *Given a dictionary D with basis B; in order to make the criteria simpler we assume that for all $k \in N \cap Y$ the variable ξ_k is not constant, i.e. depends on the variables in X (it is easy to remove this assumption). For all $k \in N \cap (X \cup Y)$ we define*

$$s_k := \begin{cases} z & \text{if } D_{zk} \neq 0, \\ \min(\{t \in B \cap X | D_{tk} \neq 0\} \cup \{k\}) & \text{otherwise;} \end{cases}$$

then: $x(D)$ is optimal for $(LP)\ |^L_{F(D)}$ for large L if and only if for all $k \in N \cap (X \cup Y)$ with $s_k = k$ or $D_{s_k k} > 0$ holds "k-bound = upperbound" and for all other $k \in N \cap (X \cup Y)$ "k-bound = lowerbound".

Proof: Consider $\xi_B = D\xi_N$: We have to discuss the influence of a change of a nonbasis variable ξ_k on the vector $(c^T x, x_1, \ldots, x_d)$ which should be maximized. $x(D)$ is optimal for $(LP)\ |^L_{F(D)}$ if and only if for every $k \in N \cap (X \cup Y)$ a possible change of ξ_k (i.e. a change where ξ_k remains feasible for h_k) does not increase (and hence decrease) the vector $(c^T x, x_1, \ldots, x_d)$; for $k \in N \setminus (X \cup Y)$ is $\xi_k \equiv 1$ fix anyway. For any $k \in N \cap (X \cup Y)$ define s_k as above.

- If $s_k = z$: A change of ξ_k does also change the value of $z = c^T x$. We can not increase z if and only if the corresponding change of ξ_k is not possible, i.e. if

$$\begin{aligned} \text{when } D_{zk} > 0: &\quad \text{"}k\text{-bound = upperbound"}, \\ \text{when } D_{zk} < 0: &\quad \text{"}k\text{-bound = lowerbound"}. \end{aligned}$$

- If $s_k \neq z$: We have to find the x_s with $s =$ the minimal index such that a change of ξ_k does also change x_s:
  - If $k \notin X$, then $\{t \in B \cap X | D_{tk} \neq 0\} \neq \emptyset$ (otherwise ξ_k would be a constant; this case was excluded in the assumptions), and since the indices in X are smaller than the indices in Y we have

$$\min(\{t \in B \cap X | D_{tk} \neq 0\} \cup \{k\}) = \min\{t \in B \cap X | D_{tk} \neq 0\}$$

    and $s_k < k$; the rest follows in the same way as for $s_k = z$.
  - If $k \in X$, then we have to remark that possibly the change of $\xi_k \equiv x_k$ itself is the most important influence on $(c^T x, x_1, \ldots, x_d)$ (hence we have to consider k when min ... is computed); the rest is similar as we have seen it above.

■

We remark that this criteria can also be useful for the design of other pivot algorithms which use a L-box. Furthermore we see that for the optimality test of the dictionary it suffices to check the signs of dictionary elements. An optimality test as in theorem 12 causes costs of $O(d^2)$, when every dictionary element is available in constant time; the total costs of the pivot search in the described manner would be $O(d) \cdot (O(dn) + O(d^2)) = O(d^2 n)$.

Instead of pivoting for every $j \in N \cap (X \cup Y)$ and applying then the criteria we better compute the effect of a pivot operation (for feasibility and maximization of $(c^T x, x_1, \ldots, x_d)$) without computing the new dictionary; it is possible to design an algorithm **MSWFindPivot**(D, i, s_i) with input D and i as in the problem of the pivot search and $s_i := \text{sign}(\xi_i^=)$, which solves the problem of the pivot search with computational costs of $O(d^2)$; the output of **MSWFindPivot** is the pivot column $j \in N \cap (X \cup Y)$ in the feasible case and $j = 0$ otherwise.

5.4 Analysis of MSWPivot

We have seen in subsection 5.2 the main ideas how to map the pivot form of the algorithm of Matoušek, Sharir, Welzl into the framework of LP-type problems; there we have an analysis (see section 3) which we like to apply also for the pivot form. Remark that by the definition of (H, v) and the mapping from D and M to G and F as in subsection 5.2 we have a relation between the two algorithms such that their behaviour is almost identical; in fact a detailed analysis of **MSWPivot** leads to the same complexity results as in **MSW**, which is new for a precise and complete LP algorithm. We present **MSWPivot** (for **MSWFindPivot** see subsection 5.3):

Algorithm MSWPivot:
Input: A set $M \subseteq X \cup Y$ of marked constraints with $N \cap (X \cup Y) \subseteq M$; a dictionary D with basis B and nonbasis N such that $x(D)$ is the optimal solution of $(LP)\,|^L_{F(D)}$ for large L.
Output: A pair $(\bar{D}, i)$ such that $\bar{D}$ is a dictionary (with basis $\bar{B}$) and $i \in \bar{B}$, where either $i = n+1$ and $x(\bar{D})$ is the optimal solution of $(LP)\,|^L_{G(M)}$ for all $L \geq L_0$ or $i \neq n+1$ such that i is violated by $\bar{D}$ and the $d+1$ constraints in $F(\bar{D}) \cup \{h_i\}$ define an infeasible linear program.

begin MSWPivot(M, D);
  if $B \cap M = \emptyset$ **then return** $(D, n+1)$
  else
    choose $i \in B \cap M$ at random;
    $(D, k) :=$ **MSWPivot**$(M \setminus \{i\}, D)$; *(first type recursive call)*
    if $k \neq n+1$ *(infeasible case)* or i not violated by D **then return** (D, k)
    else
      if the violated constraint h_i is $\xi_i \geq 0$ **then** $s_i := -1$ **else** $s_i := 1$ **endif**;
      $j :=$ **MSWFindPivot**(D, i, s_i);
      if $j = 0$ *(infeasible case)* **then return** (D, i)
      else
        pivot on (i, j);
        if $h_i \equiv \xi_i \geq 0$ **then** set flag "i-bound = lowerbound"
        else set flag "i-bound = upperbound"
        endif;
        return MSWPivot(M, D) *(second type recursive call)*
      endif
    endif
  endif
end MSWPivot.

In order to solve a linear programming problem $(LP)_{\geq 0}$ we solve $(LP)\,|^L$; we use the dictionary form (1) with initial basis $B = Y \cup \{n+1\}$; the bound flags for $i \in X$ are "i-bound = lowerbound" if $c_i < 0$, and "i-bound = upperbound" otherwise. We start **MSWPivot** with $M = X \cup Y$ and D; $F(D)$ is a basis for the corresponding LP-type optimization problem with $v(F(D)) > -\infty$. If we look in detail at the mapping from **MSWPivot** to **MSW**, we can apply the same arguments which we need in the proof of theorem 3 and obtain the following result:

Theorem 13 (Analysis of MSWPivot) *With $d = |X|$ and $n = |X| + |Y|$:*
MSWPivot(M, D) *terminates after at most $2^{n+d+1} - 2$ further calls of* **MSWPivot** *with correct output and has an expected running time of at most $e^{4 \cdot \sqrt{d \ln(n-d+1)}} \cdot O(dn)$.*

Proof: We restrict all considerations to $L \geq L_0$ (see theorem 5). Denote with $v(G)$ the value of the optimal solution of $(LP)\,|^L_G$ (similar to the definition in theorem 4). For inputs M

and D of **MSWPivot** define

$$\mathcal{E}(M,D) := \{i \in N \cap (X \cup Y) \mid v(G(M) \setminus \{h_i\}) > v(F(D))\}$$

and $n_M := |G(M)|$ and $d_{M,D} := -|\mathcal{E}(M,D)|$ (then $d \le n_M \le n+d$ and $-d \le d_{M,D} \le 0$). The case $B \cap M = \emptyset$ is trivial, so assume $B \cap M \neq \emptyset$. Order all $i \in B \cap M$ with $v(G(M \setminus \{i\})) > v(G(M))$ such that

$$v(G(M \setminus \{i_1\})) \ge v(G(M \setminus \{i_2\})) \ge \cdots \ge v(G(M \setminus \{i_s\}));$$

obviously $s \le |B \cap M|$. When we choose $i \in B \cap M$, then either $i \notin \{i_1, \ldots, i_s\}$ and hence i will not be violated by the dictionary $\hat{D}$ which is returned from the first recursive call, or $i = i_k$ for some $k \in \{1, \ldots, s\}$; if then there is a pivot step, we denote by $\tilde{D}$ the new dictionary; remark that $v(F(\hat{D})) > v(F(\tilde{D}))$. We can prove $\mathcal{E}(M,D) \subseteq \mathcal{E}(M \setminus \{i\}, D)$ and $\mathcal{E}(M,D) \subsetneq \mathcal{E}(M,\tilde{D})$, namely the variables $i_1, \ldots, i_k$ are all in $\mathcal{E}(M,\tilde{D}) \setminus \mathcal{E}(M,D)$; by this we conclude $k \le d - |\mathcal{E}(M,D)|$. An inductive argumentation in the recursion tree (over $n_M + d_{M,D}$) leads to the proof that **MSWPivot**(M,D) terminates after at most $2^{n_M + d_{M,D}+1} - 2$ further calls of **MSWPivot** with correct output. We denote by $\beta(\ell, r)$ the maximum expected number of second type recursive calls of **MSWPivot**, where the maximum is taken over all M, D where $x(D)$ is optimal solution of $(LP)\,|^L_{F(D)}$, $N \cap (X \cup Y) \subseteq M$, $r = |B \cap M|$, $d - \ell \le -d_{M,D}$ (if there are no such M and D for given ℓ and r, then we set $\beta(\ell, r) := 0$). With the results from above we can prove

$$\text{for } \ell, r \text{ with } 0 \le \ell \le d,\ 0 \le r \le n-d: \ \beta(\ell, 0) = \beta(0, r) = 0 \tag{2}$$

and

$$\text{for } \ell, r \text{ with } 1 \le \ell \le d,\ 1 \le r \le n-d: \ \beta(\ell, r) \le \beta(\ell, r-1) + \frac{1}{r} \sum_{i=1}^{\min\{\ell, r\}} (1 + \beta(\ell - i, r)); \tag{3}$$

(2) and (3) imply (for a proof see [7])

$$\text{for } \ell, r \text{ with } 0 \le \ell \le d,\ 0 \le r \le n-d: \quad \beta_p(\ell, r) \le e^{4 \cdot \sqrt{\ell \ln(r+1)}} - 1.$$

Since we have at most once a pivot search without a following call of **MSWPivot** (because then we detect infeasibility), we have for the expected number of pivot searches an upper bound of $e^{4 \cdot \sqrt{\ell \ln(r+1)}}$, where every pivot search has costs of $O(d^2)$ and is followed (if not infeasibility is detected) by a pivot operation; for the expected number violation tests we have an upper bound of $r \cdot (e^{4 \cdot \sqrt{\ell \ln(r+1)}} + 1)$, and for the remaining arithmetics we have expected costs of at most $O(r \cdot e^{4 \cdot \sqrt{\ell \ln(r+1)}})$. This completes the proof. ∎

6 Experimental Results

For a set of 50 linear programs, all of them randomly generated dual Kuhn-Quandt problems, namely with $d \in \{10, 20, \ldots, 50\}$ and $n + d \in \{100, 200, \ldots, 1000\}$, we have compared the following three algorithms: **MSWPivot**, Dual Simplex (with largest coefficient rule), and Criss-Cross (with randomized pivoting strategy). Our tests do not allow to give a detailed comment about the practical behavior of the algorithms; but we can state the main tendencies: The approximate number of pivots is for the dual simplex 20 for the smallest example and 300 for the largest, for **MSW** 100 and 6000, for Criss-Cross 300 and 750000; when we compare this to the theoretical upper bound of the expected number of pivot steps (namely $e^{4 \cdot \sqrt{d \ln(n-d+1)}}$) where the range is $6 \cdot 10^{11}$ to $2 \cdot 10^{32}$, we see that there is an enormous gap. The standard dual simplex seems to be most efficient in practice (at least for the test set as mentioned above), **MSWPivot** needs more pivot steps but works quite well, where Criss-Cross uses a lot more pivots; for our examples the standard deviation of the number of pivot steps in **MSWPivot** is rather small.

7 Final Remarks

The MSW pivot algorithm is a randomized dual simplex algorithm applied to a perturbed LP with an additional combinatorial bounding box. Dualizing the interpretation, it is a one-phase primal simplex method to a perturbed LP using a symbolic big-M technique, see [10]. Furthermore, excluding the big-M technique, it is essentially a randomized version of Bland's recursive method [1] which is the first finite algorithm for oriented matroid programming. Unlike the standard simplex method, these algorithms are combinatorial pivot algorithms, i.e. those for which the pivot choice does not depend on the magnitude of entries in the LP dictionary (tableau) but only their signs.

While the standard dual simplex method outperforms the MSW pivot algorithm and perhaps other combinatorial pivot algorithms for standard random LP problems, it is not clear at all whether any reasonably designed and randomized combinatorial pivot algorithm, e.g. the MSW pivot algorithm, runs in an expected time which is polynomial in d and n or not. The design and analysis of such strongly polynomial LP algorithms remains one of the most challenging open problems in the theory of linear programming.

Acknowledgements

The author thanks Komei Fukuda for the fruitful discussions and the helpful suggestions which made this work possible.

References

[1] R. G. Bland, New finite pivoting rules for the simplex method, *Mathematics of Operations Research*, series 2, 1977, pp. 103–107.

[2] G. B. Dantzig: *Linear Programming and Extensions,* Princeton University Press, Princeton, NJ, 1963.

[3] M. Goldwasser: A Survey of Linear Programming in Randomized Subexponential Time, *SIGACT News*, 26 no. 2, 1995, pp. 96–104.

[4] G. Kalai, A subexponential randomized simplex algorithm, In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, 1992, pp. 475–482.

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

[6] L. G. Khachiyan: Polynomial algorithms in linear programming, *U.S.S.R Comput. Maths. Math. Phys.*, Vol. 20, No. 1, 1980, pp. 53–72.

[7] 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.

[8] J. Matoušek: Lower Bounds for a Subexponential Optimization Algorithm, *Random Structures and Algorithms*, Vol. 5, No. 4, John Wiley & Sons, 1994, pp. 591–607.

[9] N. Megiddo: Linear programming in linear time when the dimension is fixed, *Journal of the Association for Computing Machinery,* 31, 1984, pp. 114–127.

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

[11] R. Seidel: Small-Dimensional Linear Programming and Convex Hulls Made Easy, *Discrete & Computational Geometry 6*, Springer-Verlag, New York, 1991, pp. 423–434.

[12] M. Sharir, E. Welzl: A combinatorial bound for linear programming and related problems, In A. Finkel, M. Jantzen, editors, *STACS 92*, volume 577 of *Lecture Notes in Computer Science,* Springer-Verlag, 1992, pp. 569–579.