An Improved Version of the Random-Facet Pivoting Rule for the Simplex Algorithm

Thomas Dueholm Hansen* Uri Zwick†

April 20, 2015

Abstract

The Random-Facet *pivoting rule* of Kalai and of Matoušek, Sharir and Welzl is an elegant randomized pivoting rule for the *simplex algorithm*, the classical combinatorial algorithm for solving *linear programs* (LPs). The expected number of pivoting steps performed by the simplex algorithm when using this rule, on any linear program involving n inequalities in d variables, is $2^{O(\sqrt{(n-d)\ \underline{\log}\,(d/\sqrt{n-d})})}$, where $\underline{\log}\, n = \max\{1, \log n\}$. A dual version of the algorithm performs an expected number of at most $2^{O(\sqrt{d\ \underline{\log}\,((n-d)/\sqrt{d})})}$ *dual* pivoting steps. This dual version is currently the fastest known combinatorial algorithm for solving general linear programs. Kalai also obtained a primal pivoting rule which performs an expected number of at most $2^{O(\sqrt{d}\ \log n)}$ pivoting steps. We present an improved version of Kalai's pivoting rule for which the expected number of primal pivoting steps is at most $\min\left\{2^{O(\sqrt{(n-d)\ \underline{\log}\,(d/(n-d))})}, 2^{O(\sqrt{d\ \underline{\log}\,((n-d)/d)})}\right\}$. This seemingly modest improvement is interesting for at least two reasons. First, the improved bound for the number of *primal* pivoting steps is better than the previous bounds for both the primal and dual pivoting steps. There is no longer any need to consider a dual version of the algorithm. Second, in the important case in which $n = O(d)$, i.e., the number of linear inequalities is linear in the number of variables, the expected running time becomes $2^{O(\sqrt{d})}$ rather than $2^{O(\sqrt{d \log d})}$. Our results, which extend previous results of Gärtner, apply not only to LP problems, but also to *LP-type* problems, supplying in particular slightly improved algorithms for solving *2-player turn-based stochastic games* and related problems.

*Department of Computer Science, Aarhus University, Denmark. Supported by The Danish Council for Independent Research | Natural Sciences (grant no. 12-126512). E-mail: tdh@cs.au.dk.

†Blavatnik School of Computer Science, Tel Aviv University, Israel. Research supported by BSF grant no. 2012338 and by The Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11). E-mail: zwick@tau.ac.il.

1 Introduction

The simplex algorithm. Linear programming (LP) [7, 11, 45, 53] is one of the most important mathematical optimization problems. The *simplex algorithm* (Dantzig [11]) is one of the most widely used methods for solving linear programs. It starts at a *vertex* of the polytope corresponding to the linear program. (We assume, for simplicity, that the linear program is feasible, bounded, and non-degenerate, and that an initial vertex of the polytope is available.) If the current vertex is not optimal, then at least one of the *edges* incident to it leads to a neighboring vertex with a larger objective value. A *pivoting rule* determines which one of these vertices to move to. The simplex algorithm, with any pivoting rule, is guaranteed to find an optimal solution of the linear program. Unfortunately, with essentially all known *deterministic* pivoting rules, the simplex method requires *exponential time* on some linear programs (see Klee and Minty [41] and [1, 2, 18, 27, 32]). While there are polynomial time algorithms for solving LP problems, most notably the *ellipsoid algorithm* (Khachian [40]) and *interior point methods* (Karmarkar [38]), these algorithms are not *strongly* polynomial, i.e., their running time, in the unit-cost model, depends on the number of *bits* in the representation of the coefficients of the LP, and not just on the combinatorial size of the problem, i.e., the number of variables and constraints. Other polynomial, but not strongly polynomial, algorithms for solving LP problems are described in [3, 12, 39]. The question whether there exists a strongly polynomial time algorithm for solving linear programs is of great theoretical importance.

Diameter of polytopes. The existence of a polynomial version of the simplex algorithm would clearly imply that the combinatorial *diameter* of each polytope is polynomial in n, the number of facets, and d, the dimension. The Hirsch conjecture (see, e.g., [11], pp. 160,168), which states that the diameter of the graph defined by an n-facet d-dimensional polytope is at most $n-d$ has recently been refuted by Santos [51, 52]. The diameter is widely believed to be polynomial in n and d, but the best known upper bound is a quasi-polynomial bound of $(n-d)^{\log_2 d}$ obtained recently by Todd [59], improving an $n^{\log_2 d+2}$ bound of Kalai and Kleitman [34, 37].

The Random-Facet pivoting rule. Kalai [33] and Matoušek, Sharir and Welzl [46] devised a *randomized* pivoting rule, Random-Facet, for the simplex algorithm and obtained a *subexponential* $2^{O(\sqrt{(n-d)\underline{\log}(d/\sqrt{n-d})})}$ upper bound on the expected number of pivoting steps it performs on *any* linear program, where d is the *dimension*, i.e., the number of variables, and n is the number of inequalities. Matoušek et al. [46] actually devised a dual version of the Random-Facet pivoting rule and obtained an upper bound of $2^{O(\sqrt{d\underline{\log}((n-d)/\sqrt{d})})}$ on the number of dual pivoting steps it performs. Random-Facet is currently the fastest known pivoting rule for the simplex algorithm. It is interesting to note that Kalai [33] and Matoušek et al. [46] used completely different routes to obtain their dual pivoting rules. Kalai's [33] pivoting rule is derived from his quasi-polynomial upper bound on the diameter of polytopes [34, 37]. Matoušek et al. [46] obtained their algorithm by improving a simple randomized linear programming algorithm of Seidel [56] which runs in $O(d!\,n)$ expected time, i.e., in linear time when the dimension d is fixed. Seidel's algorithm is an improvement over several previous algorithms [8, 13, 14, 48]. When $n \gg d$, the algorithm of Matoušek et al. [46] can be improved by combining it with an algorithm of Clarkson [9], yielding an algorithm whose complexity is $O(d^2 n + e^{O(\sqrt{d \log d})})$. We refer to Goldwasser [28] for a survey on the Random-Facet pivoting rule.

LP-type problems. Random-Facet can be used to solve a wider class of abstract optimization problems, known as *LP-type* problems [46], that includes geometrical problems such as *smallest enclosing ball* and the *distance between two polyhedra*. Various *2-player turn-based stochastic* (and

non-stochastic) *games* (2TBSGs) [10, 30, 60] can also be cast as LP-type problems [29]. For *non-discounted* games, RANDOM-FACET currently yields the fastest known algorithms [4, 5, 43].

Our results. We present an improved version of the RANDOM-FACET pivoting rule which performs an expected number of at most $\min\left\{2^{O(\sqrt{(n-d)\ \underline{\log}\,(d/(n-d))})}, 2^{O(\sqrt{d\ \underline{\log}\,((n-d)/d)})}\right\}$ primal pivoting steps. Our pivoting rule is a tuned, rigorously stated and analyzed, variant of a pivoting rule suggested by Kalai [33]. In [33], Kalai suggests three pivoting rules called S_0, S_1, and S_2. S_0 is the basic primal RANDOM-FACET referred to above. For S_1, an upper bound of $2^{O(\sqrt{d}\log n)}$ is proved. (Note that the $\log n$ here is outside the square root.) For S_2, an upper bound of $2^{O(\sqrt{d\log(n-d)})}$ is claimed. However, there are some unresolved issues with the description and analysis of S_2, issues acknowledged by Kalai [36]. In particular, steps performed by S_2 may depend on the current number of *active facets* (see Section 3), a number that is usually not available to the algorithm. Our pivoting rule IMPROVED-RANDOM-FACET is a tuned version of Kalai's pivoting rule S_2 that relies on the concept of active facets only in its analysis. In [35], Kalai suggests two other pivoting rules called "Algorithm I" and "Algorithm II". Unfortunately, these two pivoting rules suffer from similar problems, some of them documented in [50]. Gärtner [23] used Kalai's ideas to obtain a subexponential algorithm for a much wider class of problems which he calls *Abstract Optimization Problems* (AOPs). His main motivation was obtaining subexponential combinatorial algorithms for non basis-regular LP-type problems (see [23, 46]) such as computing the *minimum enclosing ball* of a set of points in $\mathbb{R}^d$ and computing the *distance between two convex polyhedra*. Our pivoting rule is a slightly improved version of Gärtner's algorithm, tuned for linear programs and other basis-regular LP-type problems.

The main contribution of this paper is a rigorous derivation and analysis of a pivoting rule, IMPROVED-RANDOM-FACET, which yields results that are slightly stronger than those claimed by Kalai [33, 35]. In particular, we obtain the fastest known combinatorial algorithm for solving linear programs, the first improvement in over 20 years. The improvement is particularly significant when $n = O(d)$, i.e., when the number of inequality constraints is linear in the number of variables, in which case the expected number of pivoting steps is reduced from $2^{O(\sqrt{d\log d})}$ to $2^{O(\sqrt{d})}$. For larger values of n, the improvement is only by a constant factor in the exponent. It is also worth mentioning that the improved algorithm is now a purely primal algorithm.

We believe that the improved pivoting rule is essentially the best pivoting rule that can be obtained using the (simple and ingenious) idea of choosing a random facet. As a step towards designing IMPROVED-RANDOM-FACET, we consider a hypothetical "pivoting rule" IDEAL-RANDOM-FACET which is allowed to sample, at no cost, an active facet. (A similar idea was used by Gärtner [23].) IDEAL-RANDOM-FACET, which is not an implementable pivoting rule, plays a central role in both the design and the analysis of IMPROVED-RANDOM-FACET. In particular, we show that IMPROVED-RANDOM-FACET essentially matches the (unrealizable) performance of IDEAL-RANDOM-FACET. We thus believe that new ingredients are required to obtain further improved results.

The analysis of IDEAL-RANDOM-FACET gives rise to an interesting 2-dimensional recurrence relation. The technically challenging analysis of this recurrence relation may also be viewed as one of the contributions of this paper.

Throughout this paper we adopt the primal point of view. For concreteness, we consider linear programming problems. All our algorithms can be used, with minor modifications, to solve more general basis-regular LP-type problems. In particular, we obtain slightly improved algorithms for various classes of 2TBSGs.

Lower bounds. In [19, 20, 22], building on results of Friedmann [17] and Fearnley [15], we showed that RANDOM-FACET may require an expected number of $2^{\tilde{\Omega}(n^{1/3})}$ steps even on linear programs

that correspond to *shortest paths* problems. (In [20] we obtained an $2^{\tilde{\Omega}(n^{1/2})}$ lower bound for a one-permutation variant of RANDOM-FACET that we thought was equivalent to RANDOM-FACET. Embarrassingly, as we point out in [21], the two pivoting rules are *not* equivalent.) Matoušek [44] obtained an essentially tight $2^{\Omega(n^{1/2})}$ lower bound on the complexity of RANDOM-FACET on AUSOs, which form a special class of LP-type problems. It is an intriguing open problem whether similar lower bounds can be proved for IMPROVED-RANDOM-FACET.

In [20] we also obtained an $2^{\tilde{\Omega}(n^{1/4})}$ lower bound on the number of pivoting steps performed by RANDOM-EDGE on linear programs that correspond to Markov Decision Processes (MDPs) [49]. Matoušek and Szabó [47] obtained an $2^{\tilde{\Omega}(n^{1/3})}$ lower bound on the complexity of RANDOM-EDGE on AUSOs. RANDOM-EDGE is perhaps the simplest randomized pivoting rule. If there are several edges leading to vertices with larger objective value, pick one of them uniformly at random. Only exponential upper bounds are known for RANDOM-EDGE [25, 31].

Acyclic unique sink orientations (AUSOs). Acyclic unique sink orientations (AUSOs) of n-cubes [26, 54, 55, 58] provide an elegant combinatorial abstraction of linear programs and other computational problems. RANDOM-FACET, as analyzed by Gärtner [24], is currently the fastest known algorithm for finding the sink of an AUSO. It finds the sink of any AUSO of an n-cube using an expected number of at most $e^{2\sqrt{n}}$ steps. Matoušek [44] supplies an almost matching lower bound. We believe that a specialized analysis of IMPROVED-RANDOM-FACET for combinatorial cubes can improve Gärtner's [24] bound for AUSOs.

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we give a brief introduction to linear programming and the simplex algorithm. In Section 3 we review the RANDOM-FACET pivoting rule of Kalai [33] and Matoušek et al. [46]. In Section 5 we describe the IDEAL-RANDOM-FACET "pivoting rule". Some of the technical details from Section 5 appear in appendices A and B. In Section 6, which is the main section of this paper, we describe our improved pivoting rule IMPROVED-RANDOM-FACET. We end in Section 7 with some concluding remarks and open problems.

2 Linear programming and the simplex algorithm

We begin with a brief description of linear programming and the simplex algorithm, allowing us to explain the terminology and notation used throughout the paper.

A linear programming problem in $\mathbb{R}^d$ is defined by a set F of linear *inequality constraints* and by an *objective vector* $\mathbf{c} \in \mathbb{R}^d$. Each $f \in F$ is an inequality of the form $\mathbf{a}_f^T\mathbf{x} \leq b_f$, where $\mathbf{a}_f \in \mathbb{R}^d$, $b_f \in \mathbb{R}$, and $\mathbf{x} \in \mathbb{R}^d$. Let $P(F) = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{a}_f^T\mathbf{x} \leq b_f \text{ for } f \in F\}$ be the *polyhedron* defined by F. The objective is to find a point $\mathbf{x} \in P(F)$ which maximizes the objective function $\mathbf{c}^T\mathbf{x}$. A linear program is *feasible* if $P(F) \neq \emptyset$, and *bounded* if the objective function is bounded from above.

A set $B \subseteq F$, $|B| = d$, is said to be a (feasible) *basis*, if there is a unique solution $\mathbf{x} = \mathbf{x}(B)$ of the d linear equations $\mathbf{a}_f^T\mathbf{x} = b_f$, for $f \in B$, and if $\mathbf{a}_f^T\mathbf{x} \leq b_f$, for $f \in F$. The point $\mathbf{x}(B)$ is said to be a *vertex* of $P(F)$. The constraints $f \in B$ are said to be *facets*. Let $v(B) = \mathbf{c}^T\mathbf{x}(B)$. It is well known that the optimum of a feasible and bounded linear program is always attained at at least one vertex. A linear program is said to be *non-degenerate* if each vertex is defined by a unique basis and if no two vertices have the same objective value. We restrict our attention in this paper to feasible, bounded, non-degenerate problems, the optimum of which is attained at a unique vertex. The non-degeneracy assumption can be removed using standard techniques (see, e.g., [7, 45]).

If B is a basis and $f \in B$, then $B \setminus \{f\}$ defines an *edge* of $P(F)$ that connects the vertex $\mathbf{x}(B)$ with a vertex $\mathbf{x}(B')$, where $B' = B \setminus \{f\} \cup \{f'\}$, for some $f' \in F$. (If the polyhedron is unbounded

```
Algorithm RANDOM-FACET(F, B_0)
  if F ∩ B_0 = ∅ then
    return B_0
  else
    f ← RANDOM(F ∩ B_0)
    B_1 ← RANDOM-FACET(F \ {f}, B_0)
    B_2 ← PIVOT(F, B_1, f)
    if B_1 ≠ B_2 then
      return RANDOM-FACET(F, B_2)
    else
      return B_1
```

Figure 1: The simplex algorithm with the RANDOM-FACET pivoting rule.

from below, $B \setminus \{f\}$ may define a *ray* rather than an edge.) If $v(B') > v(B)$ we refer to the move from $\mathbf{x}(B)$ to $\mathbf{x}(B')$ as a *pivot* step in which f leaves the basis and f' enters it. It is well known that $\mathbf{x}(B)$ is the optimal vertex if and only if no (improving) pivot step can be made from it.

The simplex algorithm [11] starts at some initial vertex $\mathbf{x}(B_0)$. (Finding an initial vertex usually requires the solution of an auxiliary linear program for which an initial vertex is known.) As long as the current vertex is not optimal, a pivot step is performed. If more than one pivot step is possible, a *pivoting rule* is used to select the pivot step to be performed. As the number of vertices is finite, the simplex algorithm, with any pivoting rule, always terminates after a finite number of steps.

We assumed so far that all the constraints of the linear program are inequality constraints. The algorithms we consider occasionally convert inequality constraints to equality constraints. We find it convenient to leave these equality constraints in B and remove them from F. We thus view (F, B) as a specification of a linear program, along with an initial basis, in which F is the set of inequality constraints and $B \setminus F$ is the set of equality constraints. Thus, only constraints from $F \cap B$ are allowed to leave B when performing a pivot step. Also note that if $|F \cap B| = d'$, then by using the equality constraints of $B \setminus F$ to eliminate some of the variables, we can convert the linear program into an equivalent linear program involving only d' variables.

3 The RANDOM-FACET pivoting rule

Inspired by the quasi-polynomial upper bound on the *diameter* of polytopes (see [34, 37, 59]), Kalai [33, 35] proposed several randomized pivoting rules. The simplest one of them is the following. (As mentioned, a dual version of this pivoting rule was suggested by Matoušek et al. [46].)

Let $\mathbf{x}(B_0)$ be the current vertex visited by the simplex algorithm. Randomly choose one of the facets $f \in F \cap B_0$. Recursively find the optimal vertex $\mathbf{x}(B_1)$ among all vertices of the polytope that lie on f. (Note that $f \in B_1$.) This corresponds to finding an optimal solution of a linear program of dimension $d-1$ in which the inequality corresponding to f is replaced by an equality. If $\mathbf{x}(B_1)$ is also an optimal solution of the original problem, we are done. Otherwise, assuming non-degeneracy, the only pivot step that can lead from $\mathbf{x}(B_1)$ to a vertex $\mathbf{x}(B_2)$ with a higher objective value is the step in which f leaves the basis, i.e., $B_2 = B_1 \setminus \{f\} \cup \{f'\}$, for some $f' \in F$. This pivot step is made and the algorithm is run recursively from $\mathbf{x}(B_2)$.

A more formal description of the algorithm is given in Figure 1. RANDOM-FACET is a recursive

procedure that receives two parameters, F is the set of *inequality constraints* that define the linear program to be solved by the current recursive call, and B_0 is the set of *equality constraints* that define the current *vertex* $\mathbf{x}_0 = \mathbf{x}(B_0)$. A problem (F, B_0) is said to be of dimension $d = |F \cap B_0|$.

If $F \cap B_0 = \emptyset$, then B_0 is the optimal solution of the problem. If this is not the case, RANDOM-FACET(F, B_0) uses the function RANDOM to choose a *random facet* $f \in F \cap B_0$ uniformly at random. It then solves recursively the problem defined by the pair $(F \setminus \{f\}, B_0)$. As explained, removing f from F, while keeping it in B_0, converts f into an equality constraint. The recursive call RANDOM-FACET$(F \setminus \{f\}, B_0)$ thus returns an optimal solution B_1 of the problem in which f is required to be satisfied with equality. If B_1 is also an optimal solution of the original problem, we are done. PIVOT(F, B_1, f) tries to perform a pivoting step from B_1 in which f leaves the basis. If the pivoting step is successful, i.e., if the edge defined by $B_1 \setminus \{f\}$ leads to a vertex with a higher objective value, the new basis $B_2 \neq B_1$, is returned by PIVOT(F, B_1, f), otherwise B_1 is returned. If $B_1 \neq B_2$, a second recursive call on the problem defined by the pair (F, B_2) is performed. (As we shall see, the second recursive call can also be made on $(F \setminus \{f\}, B_2)$ as f would never enter the basis again.)

As the simplex algorithm with the RANDOM-FACET pivoting rule is a specialization of the generic simplex algorithm, the algorithm always finds an optimal vertex after a finite number of steps, no matter what random choices were made. The random choices ensure, as we shall see, that the expected number of pivoting steps performed by the algorithm is *sub-exponential*.

The analysis of RANDOM-FACET and its variants hinges on the following definition of Kalai [33]:

Definition 3.1 (Active facets) *A constraint $f \in F$ is said to be an* active *facet of (F, B), if and only if there is a basis B' such that $f \in B'$, $v(B) \leq v(B')$, and $B \setminus F \subseteq B'$.*

We let $v_{(F,B)}(f) = v(f)$, for $f \in F$, be the value of the highest vertex that lies on f, i.e., the optimal value of the linear program $(F \setminus \{f\}, B)$ in which f is required to be satisfied with equality. Clearly, f is an active facet of (F, B) if and only if $v(f) \geq v(B)$.

Let $\bar{f}_\dagger(d, n)$, for $0 \leq d \leq n$, be the function defined by the following recurrence relation:

$$\bar{f}_\dagger(d,n) \;=\; \bar{f}_\dagger(d-1,n-1) + \frac{1}{d}\sum_{i=1}^{\min\{d,n-d\}} (1 + \bar{f}_\dagger(d,n-i))$$

for $0 < d < n$, and

$$\begin{aligned} \bar{f}_\dagger(d,d) &= 0\ , \text{ for } d \geq 0\ , \\ \bar{f}_\dagger(0,n) &= 0\ , \text{ for } n \geq 0\ . \end{aligned}$$

Theorem 3.2 ([33]) *For every $0 \leq d \leq n$, $\bar{f}_\dagger(d, n)$ is an upper bound on the expected number of pivoting steps performed by the* RANDOM-FACET *pivoting rule on any problem of dimension d with at most n active constraints with respect to the initial basis.*

Proof: Let (F, B_0) be the input to RANDOM-FACET, let $d = |F \cap B_0|$ and let n be the corresponding number of active facets. Recall that for every facet $f \in F$, $v(f)$ is the highest value of a vertex that lies on f. By definition, all the d facets of $F \cap B_0$ are active. Suppose that $F \cap B_0 = \{f_1, f_2, \ldots, f_d\}$, where $v(f_1) \leq v(f_2) \leq \cdots \leq v(f_d)$. Let $f_i \in F \cap B_0$ be the facet chosen randomly by the call on (F, B_0). By induction, the expected number of pivoting steps performed by the first recursive call on $(F \setminus \{f_i\}, B_0)$ is at most $\bar{f}_\dagger(d-1, n-1)$. This recursive call returns a basis B_1 whose objective function is $v(f_i)$. If the edge $B_0 \setminus \{f_i\}$ does not lead to a better vertex, then B_1 is also optimal for the original problem and no further pivoting steps are performed. Otherwise, PIVOT(F, B_1, f_i) returns a basis $B_2 = B_1 \setminus \{f_i\} \cup \{f'\}$ such that $v(B_2) > v(B_1) = v(f_i)$. It follows that $f_1, f_2, \ldots, f_i$

are not active facets of (F, B_2). Thus, the number of active facets of (F, B_2) is at most $n - i$. By induction, the expected number of pivoting steps made by the recursive call on (F, B_2) is at most $\bar{f}_\dagger(d, n - i)$. As i is uniformly random among $\{1, 2, \ldots, d\}$, we get the claim of the lemma. □

Let $f_\dagger(d, m) = \bar{f}_\dagger(d, d + m) + 1$. [1] By Theorem 3.2, $f_\dagger(d, m)$ is an upper bound on the number of pivoting steps performed by RANDOM-FACET, counting also the final failed pivoting step that ascertains the optimality of the solution found, on a problem of dimension d with $d + m$ active constraints. The function $f_\dagger(d, m)$ satisfies the following recurrence relation:

$$f_\dagger(d, m) \;=\; f_\dagger(d - 1, m) + \frac{1}{d} \sum_{i=1}^{\min\{d,m\}} f_\dagger(d, m - i)$$

for $d, m > 0$, and

$$f_\dagger(d, 0) \;=\; f_\dagger(0, m) \;=\; 1 \;, \text{ for } d, m \geq 0 \;.$$

Matoušek, Sharir and Welzl [46] obtained a dual version of the RANDOM-FACET algorithm. Their analysis hinges on the concept of *hidden dimension* which is dual to the concept of active facets used by Kalai [33]. Matoušek et al. [46] obtained the following tight bounds on $f_\dagger(d, m)$.

Theorem 3.3 ([46]) *For all $d, m \geq 0$,*

$$f_\dagger(d, m) = \mathrm{e}^{(2+o(1))\sqrt{m \,\underline{\log}\,(d/\sqrt{m})} + O(\sqrt{m} + \log d)} \;.$$

4 The RANDOM-FACET recurrence for combinatorial cubes

The recurrence for the RANDOM-FACET pivoting rule can be significantly simplified when the polyhedron of the linear program is a hypercube. In this case, eliminating an active facet reduces the dimension by one. We next summarize results of Gärtner [23, 24] regarding the analysis of a univariate recurrence relation that appears in the analysis of RANDOM-FACET on hypercubes, and more generally on AUSOs. For completeness, we include proofs. The recurrence is parameterized by a parameter c that will be used later:

Definition 4.1 *Let $f_c(k)$, for $k \geq 0$, be defined as*

$$\begin{aligned} f_c(k) &= f_c(k-1) + \frac{c}{k} \sum_{i=0}^{k-1} f_c(i) \quad , \quad k > 0 \\ f_c(0) &= 1 \end{aligned}$$

It is interesting to note that $f_1(k)$ is the expected number of increasing subsequences, of all possible lengths, of a random permutation on $\{1, 2 \ldots, k\}$ [16, 42]. This correspondence is also reflected in the following lemma.

Lemma 4.2 *For every $k \geq 0$,*

$$f_c(k) \;=\; \sum_{i=0}^{k} \frac{c^i}{i!} \binom{k}{i} \;\leq\; \sum_{i=0}^{k} \frac{(ck)^i}{(i!)^2} \;.$$

[1] Throughout the paper we adopt the convention that a function with a bar in its name, e.g., $\bar{f}_\dagger(d, n)$, is a function that takes as an argument the number of active constraints n, while a function without a bar, e.g., $f_\dagger(d, m)$, is a function that takes as an argument the *excess* number of constraints, i.e., $m = n - d$.

Proof: We prove the claim by induction, relying on the identity $\sum_{j=i-1}^{k-1}\binom{j}{i-1} = \frac{k}{i}\binom{k-1}{i-1}$ that can be easily proved by induction. For $k = 0$, the claim clearly holds. For $k > 0$ we get

$$\begin{aligned} f_c(k) &= f_c(k-1) + \frac{c}{k}\sum_{i=0}^{k-1} f_c(i) \\ &= \sum_{i=0}^{k-1}\frac{c^i}{i!}\binom{k-1}{i} + \frac{c}{k}\sum_{j=0}^{k-1}\sum_{i=0}^{j}\frac{c^i}{i!}\binom{j}{i} . \end{aligned}$$

The coefficient of $\frac{c^i}{i!}$ in the above expression is

$$\binom{k-1}{i} + \frac{i}{k}\sum_{j=i-1}^{k-1}\binom{j}{i-1} = \binom{k-1}{i} + \binom{k-1}{i-1} = \binom{k}{i}$$

as required. The upper bound follows easily as $\binom{k}{i} \le \frac{k^i}{i!}$. □

To upper bound $f_c(k)$ we use the following simple lemma:

Lemma 4.3 *For every $a \ge 0$ and every integer $k \ge 0$,*

$$\sum_{i=0}^{k}\frac{a^i}{(i!)^2} \le e^{2\sqrt{a}} .$$

Proof: As in the proof of Corollary 4.5 in Gärtner [23], we use $e^x = \sum_{k=0}^{\infty}\frac{x^k}{k!}$ to get:

$$\sum_{i=0}^{k}\frac{a^i}{(i!)^2} = \sum_{i=0}^{k}\left(\frac{\sqrt{a}^i}{i!}\right)^2 \le \left(\sum_{i=0}^{k}\frac{\sqrt{a}^i}{i!}\right)^2 \le \left(\sum_{i=0}^{\infty}\frac{\sqrt{a}^i}{i!}\right)^2 = e^{2\sqrt{a}} .$$

□

Combining Lemmas 4.2 and 4.3 we get:

Corollary 4.4 *For every $k \ge 0$, $f_c(k) \le \mathrm{e}^{2\sqrt{ck}}$.*

It is also not difficult to prove an almost matching lower bound for $f_c(k)$:

Lemma 4.5 *For every $k \ge 0$ and $1 \le c \le k$,*

$$f_c(k) \ge \frac{\mathrm{e}^{2\sqrt{ck}}}{\mathrm{e}^{3+c}(1+\sqrt{ck})} .$$

Proof: Let $r = \left\lceil\sqrt{ck}\right\rceil$. It follows from Lemma 4.2 that

$$f_c(k) = \sum_{i=0}^{k}\frac{c^i}{i!}\binom{k}{i} \ge \frac{c^r}{r!}\binom{k}{r} .$$

We use the fact that $1 - x \ge \mathrm{e}^{-2x}$ for $x \le \frac{1}{2}$ and get:

$$\frac{\binom{k}{r}}{\frac{k^r}{r!}} = \frac{k!}{k^r(k-r)!} = \prod_{i=1}^{r-1}\left(1-\frac{i}{k}\right) \ge \prod_{i=1}^{r-1}\mathrm{e}^{\frac{-2i}{k}} = \mathrm{e}^{\frac{-2}{k}\sum_{i=1}^{r-1} i} = \mathrm{e}^{\frac{-r(r-1)}{k}} \ge \frac{1}{\mathrm{e}^{1+c}} .$$

Using $r! \leq \mathrm{e}\sqrt{r}(r/\mathrm{e})^r$ we conclude that:

$$f_c(k) \;\geq\; \frac{c^r}{r!}\binom{k}{r} \;\geq\; \frac{(ck)^r}{\mathrm{e}^{1+c}(r!)^2} \;\geq\; \frac{\mathrm{e}^{2r}}{\mathrm{e}^{3+c}r} \;\geq\; \frac{\mathrm{e}^{2\sqrt{ck}}}{\mathrm{e}^{3+c}(1+\sqrt{ck})} \;.$$

□

One of the equivalent definitions of the *modified Bessel function of the first kind* is $I_0(z) = \sum_{i\geq 0} \frac{(\frac{1}{4}z^2)^i}{(i!)^2}$. It is thus clear that $f_c(k) \leq I_0(2\sqrt{ck})$. It is known that for $|\arg z| < \frac{\pi}{2}$, we have

$$I_0(z) \;=\; \frac{\mathrm{e}^z}{\sqrt{2\pi z}}\left(1 + \frac{1}{8z} + \frac{1\cdot 9}{2!(8z)^2} + \frac{1\cdot 9\cdot 25}{3!(8z)^3} + \ldots\right) \;.$$

Thus, a more accurate bound on the asymptotic behavior of $f_c(k)$ is $f_c(k) \sim \frac{\mathrm{e}^{2\sqrt{ck}}}{2\sqrt{\pi}(ck)^{1/4}}$ [16, 42].

5 The Ideal-Random-Facet "pivoting rule"

When $d \ll n$, the efficiency of the basic Random-Facet algorithm of Section 3 is hampered by the fact that the random facet is chosen from among the d facets containing the current vertex, rather than among all n active facets. Indeed, it is not immediately clear how the algorithm can identify more active facets, with appropriate vertices on them, to choose from. Before describing in the next section how this can be done, we discuss in this section an idealized, and seemingly unimplementable, version of the Random-Facet algorithm that can magically sample, at no cost, a uniformly random active facet, with an appropriate vertex on it.

Suppose that (F, B_0) is the current instance and that $f_1, f_2, \ldots, f_n \in F$ are the active facets of (F, B_0) such that $v(f_1) \leq v(f_2) \leq \cdots \leq v(f_n)$. If we could sample a random facet $f = f_i$ uniformly at random from $\{f_1, f_2, \ldots, f_n\}$, and obtain for free a basis B_f such that $f \in B_f$ and $v(B_0) \leq v(B_f)$, then a recursive call on $(F\setminus\{f\}, B_f)$ would return a basis B_1 such that $f_1, f_2, \ldots, f_i$ are no longer active facets of (F, B_1). As i is now uniformly random among $\{1, 2, \ldots, n\}$, rather than $\{1, 2, \ldots, d\}$, progress is expected to be much faster than before. We refer to this as the Ideal-Random-Facet "pivoting rule". A similar concept appears in Gärtner [23]. In general, a less ideal, but still unimplementable pivoting rule samples from a fraction of the active facets, for instance from among $\lceil \frac{n}{c} \rceil$ facets, for some $c \geq 1$.

Let $\bar{f}_c(d, n)$ be a function defined by the following recurrence relation:

$$\bar{f}_c(d, n) \;=\; \bar{f}_c(d-1, n) + \frac{c}{n}\sum_{i=1}^{n-d}(1 + \bar{f}_c(d, n-i)) \;,$$

for $0 < d < n$, with the initial conditions

$$\begin{aligned} \bar{f}_c(d, n) \;&=\; 0 \;,\; \text{for } n \leq d \;, \\ \bar{f}_c(0, n) \;&=\; 0 \;,\; \text{for } n \geq 0 \;. \end{aligned}$$

Using the arguments from the proof of Theorem 3.2, we get that $\bar{f}_c(d, n)$ is an upper bound on the expected number of pivoting steps performed by the Ideal-Random-Facet "algorithm" with parameter c on any problem in dimension d with at most n active constraints.

We again apply the transformation $f_c(d, m) = \bar{f}_c(d, d+m) + 1$ and get

$$f_c(d, m) \;=\; f_c(d-1, m) + \frac{c}{d+m}\sum_{j=0}^{m-1} f_c(d, j) \;,\; d, m > 0 \;,$$

$$f_c(d,0) \;=\; f_c(0,m) \;=\; 1 \quad , \quad d,m \geq 0 \; .$$

The function $f_c(d,m)$ has some remarkable properties, especially for $c = 1$. The bivariate recurrence relation for $f_c(d,m)$ is much harder to solve than the univariate recurrence relation for $f_c(k)$ (Definition 4.1). The following lemma, however, establishes an interesting relation between $f_c(k)$ and $f_c(d,m)$ that allows us to tightly bound $f_c(d,d)$.

Lemma 5.1 $f_c(k) \;=\; 1 + c\sum_{i=1}^{k} f_c(i,k-i)$, *for* $k \geq 0$.

Proof: The proof is by induction on k. For $k = 0$ we indeed have $f(0) = f(0,0) = 1$. For $k > 0$, we get using the induction hypothesis and the definitions of $f_c(k)$ and $f_c(d,m)$ (see more explanations below):

$$\begin{aligned}
f_c(k) \;&=\; f_c(k-1) + \frac{c}{k}\sum_{j=0}^{k-1} f_c(j) \\
&=\; \left(1 + c\sum_{i=1}^{k-1} f_c(i,k-1-i)\right) + \frac{c}{k}\sum_{j=0}^{k-1}\left(1 + c\sum_{i=1}^{j} f_c(i,j-i)\right) \\
&=\; 1 + c + \left(c\sum_{i=1}^{k-1} f_c(i,k-1-i)\right) + \frac{c^2}{k}\sum_{j=1}^{k-1}\sum_{i=1}^{j} f_c(i,j-i) \\
&=\; 1 + c + \left(c\sum_{i=1}^{k-1} f_c(i,k-1-i)\right) + \frac{c^2}{k}\sum_{i=1}^{k-1}\sum_{j=i}^{k-1} f_c(i,j-i) \\
&=\; 1 + c + \left(c\sum_{i=1}^{k-1} f_c(i,k-1-i)\right) + \frac{c^2}{k}\sum_{i=1}^{k-1}\sum_{j=0}^{k-i-1} f_c(i,j) \\
&=\; 1 + c + c\sum_{i=1}^{k-1}\left(f_c(i,k-1-i) + \frac{c}{k}\sum_{j=0}^{k-i-1} f_c(i,j)\right) \\
&=\; 1 + c + c\sum_{i=1}^{k-1} f_c(i,k-i) \\
&=\; 1 + c\sum_{i=1}^{k} f_c(i,k-i)
\end{aligned}$$

The first line follows from the definition of $f_c(d)$. The second line follows by induction. In the third line, 1 and c are extracted from the first and second term, respectively. This allows us to sum from $j = 1$ in the second term. The fourth line changes the order of summation in the second term, and the fifth line subtracts i from j in the second sum of the second term. The sixth line joins the two sums, the seventh line uses the definition of $f_c(i,k-i)$, and finally the eighth line uses the fact that $f_c(k,0) = 1$. □

Combining Lemma 4.2 and Lemma 5.1, we get

Corollary 5.2 $f_c(d,d) \;\leq\; f_c(2d) \;\leq\; \mathrm{e}^{2\sqrt{2cd}}$.

Lemma 5.3 *For all $d, m \geq 0$,*

$$f_c(d,m) \;=\; 1 + c \sum_{i=0}^{d-1} \sum_{j=0}^{m-1} \frac{f_c(i,j)}{i+j+2} \prod_{t=j+1}^{m-1} \frac{i+t+1+c}{i+t+2} \,.$$

When $c = 1$ the product in Lemma 5.3 simplifies to 1. Although it is not at all immediate from the definition of $f_c(d,m)$, it thus turns out that $f_1(d,m) = f_1(m,d)$, for every $d, m \geq 0$.

Lemmas 5.3 is proved by induction, and the proof essentially boils down to repeated use of the definition of $f_c(d,m)$ to expand every function call that takes either d or m as an argument. Since the proof is somewhat involved it has been deferred to Appendix A.

Lemma 5.4 *For all $d, m \geq 0$,*

$$f_c(d,m) \;=\; \sum_{\substack{1 \leq d_1 < \cdots < d_k \leq d \\ 1 \leq m_1 < \cdots < m_{k+1} = m+1}} \prod_{i=1}^{k} \frac{c}{d_i + m_i} \prod_{t=m_i+1}^{m_{i+1}-1} \left(1 + \frac{c-1}{d_i + t}\right).$$

where the sum is over all pairs of sequences $(d_1, d_2, \ldots, d_k)$ and $(m_1, m_2, \ldots, m_k)$, where $1 \leq d_1 < \cdots < d_k \leq d$ and $1 \leq m_1 < \cdots < m_k \leq m$, for $k = 0, 1, \ldots, \min\{d, m\}$. For $k = 0$, we interpret the sum to include a term corresponding to an empty sequence of d_i's and an empty sequence of m_i's. The empty product is defined, as usual, to be 1.

Proof: The proof is of course by induction on d and m. If $d = 0$ or $m = 0$, then the sum includes only a 1 term corresponding to the empty sequences, which matches the definition of $f(d,m)$.

Assume now, that the lemma holds for every (i,j), where $0 \leq i < d$ and $0 \leq j < m$. Using Lemma 5.3 and the induction hypothesis we then have (see more explanations below):

$$\begin{aligned}
f_c(d,m) \;&=\; 1 + \sum_{i=0}^{d-1} \sum_{j=0}^{m-1} \frac{c\, f_c(i,j)}{i+j+2} \prod_{t=j+1}^{m-1} \frac{i+t+1+c}{i+t+2} \\
&=\; 1 + \sum_{i=0}^{d-1} \sum_{j=0}^{m-1} \left(\frac{c}{i+j+2} \prod_{t=j+2}^{m} \left(1 + \frac{c-1}{i+t+1}\right)\right) \\
&\qquad \left(\sum_{\substack{1 \leq d_1 < \cdots < d_k \leq i \\ 1 \leq m_1 < \cdots < m_{k+1} = j+1}} \prod_{s=1}^{k} \frac{c}{d_s + m_s} \prod_{t=m_s+1}^{m_{s+1}-1} \left(1 + \frac{c-1}{d_s + t}\right)\right) \\
&=\; 1 + \sum_{i=0}^{d-1} \sum_{j=0}^{m-1} \left(\sum_{\substack{1 \leq d_1 < \cdots < d_k < d_{k+1} = i+1 \\ 1 \leq m_1 < \cdots < m_{k+1} = j+1 < m_{k+2} = m+1}} \prod_{s=1}^{k+1} \frac{c}{d_s + m_s} \prod_{t=m_s+1}^{m_{s+1}-1} \left(1 + \frac{c-1}{d_s + t}\right)\right) \\
&=\; \sum_{\substack{1 \leq d_1 < \cdots < d_{k+1} \leq d \\ 1 \leq m_1 < \cdots < m_{k+2} = m+1}} \prod_{s=1}^{k+1} \frac{c}{d_s + m_s} \prod_{t=m_s+1}^{m_{s+1}-1} \left(1 + \frac{c-1}{d_s + t}\right)
\end{aligned}$$

In the first line we simply used the definition of $f_c(d,m)$. In the second line we used the induction hypothesis, and we shifted t by 1 in the first product. In the third line, we extend each sequence

$(d_1, d_2, \ldots, d_k)$, in which $d_k \le i$, into a sequence $(d_1, d_2, \ldots, d_k, d_{k+1})$, where $d_{k+1} = i+1$, and similarly each $(m_1, m_2, \ldots, m_k)$ into $(m_1, m_2, \ldots, m_k, m_{k+1})$, where $m_{k+1} = j+1$. When we sum up over all i and j we sum up over all pairs of sequences, getting the fourth line. □

A somewhat similar "closed-form" for $f_\dagger(d,m)$ is given by Matoušek et al. [46] (Lemma 7). Using this "closed-form" we prove asymptotically matching upper and lower bounds on $f_c(d,m)$:

Lemma 5.5 *For all $d, m \ge 1$ and constant $c \ge 1$*

$$\begin{aligned} f_c(d,m) &\le m^{c-1}\, \mathrm{e}^{2\sqrt{cd}\left(\sqrt{2}+\sqrt{\ln \frac{m}{d}}\right)}\ , \quad \text{if } d \le m\ , \\ f_c(d,m) &\le m^{c-1}\, \mathrm{e}^{2\sqrt{cm}\left(\sqrt{2}+\sqrt{\ln \frac{d}{m}}\right)}\ , \quad \text{if } d > m\ . \end{aligned}$$

In both cases

$$f_c(d,m) = \min\left\{2^{O(\sqrt{d \log \frac{m}{d}})}, 2^{O(\sqrt{m \log \frac{d}{m}})}\right\}\ .$$

Proof: The upper bound is obtained using the "closed-form" of Lemma 5.4. Observe first that:

$$\begin{aligned} f_c(d,m) &= \sum_{\substack{1\le d_1<\cdots<d_k\le d \\ 1\le m_1<\cdots<m_{k+1}=m+1}} \prod_{i=1}^{k} \frac{c}{d_i+m_i} \prod_{t=m_i+1}^{m_{i+1}-1}\left(1+\frac{c-1}{d_i+t}\right) \\ &\le \left(\sum_{\substack{1\le d_1<\cdots<d_k\le d \\ 1\le m_1<\cdots<m_k\le m}} \prod_{i=1}^{k} \frac{c}{d_i+m_i}\right)\left(\prod_{t=3}^{m}\left(1+\frac{c-1}{t}\right)\right) \end{aligned}$$

We bound the two factors separately.

$$\prod_{t=3}^{m}\left(1+\frac{c-1}{t}\right) < \prod_{t=3}^{m} \mathrm{e}^{\frac{c-1}{t}} = \mathrm{e}^{\sum_{t=3}^{m}\frac{c-1}{t}} = \mathrm{e}^{(c-1)(H_m-\frac{3}{2})} \le \mathrm{e}^{(c-1)\ln m} = m^{c-1}$$

To upper bound

$$\sum_{\substack{1\le d_1<\cdots<d_k\le d \\ 1\le m_1<\cdots<m_k\le m}} \prod_{i=1}^{k} \frac{c}{d_i+m_i}$$

we first observe that the sum is symmetric in d and m. For the remainder of the proof we therefore assume that $d \le m$. The next idea is to split every pair of sequences $1 \le d_1 < \cdots < d_k \le d$ and $1 \le m_1 < \cdots < m_k \le m$ into two parts $1 \le d_1 < \cdots < d_s \le d$ and $1 \le m_1 < \cdots < m_s \le d$, and $1 \le d'_1 < \cdots < d'_r \le d$ and $d < m'_1 < \cdots < m'_r \le m$.

$$\begin{aligned} \sum_{\substack{1\le d_1<\cdots<d_k\le d \\ 1\le m_1<\cdots<m_k\le m}} \prod_{i=1}^{k} \frac{c}{d_i+m_i} &\le \sum_{\substack{1\le d'_1<\cdots<d'_r\le d \\ d<m'_1<\cdots<m'_r\le m}} \prod_{i=1}^{r} \frac{c}{m'_i} \sum_{\substack{1\le d_1<\cdots<d_s\le d \\ 1\le m_1<\cdots<m_s\le d}} \prod_{i=1}^{s} \frac{c}{d_i+m_i} \\ &\le f_c(d,d)\cdot \sum_{r=0}^{d}\binom{d}{r} \sum_{d<m'_1<\cdots<m'_r\le m} \prod_{i=1}^{r} \frac{c}{m'_i} \\ &\le f_c(d,d)\cdot \sum_{r=0}^{d}\binom{d}{r}\frac{1}{r!}\left(\sum_{j=d+1}^{m}\frac{c}{j}\right)^r \end{aligned}$$

$$
\begin{aligned}
&\leq f_c(d,d) \cdot \sum_{r=0}^{d} \frac{1}{(r!)^2} \left(cd \sum_{j=d+1}^{m} \frac{1}{j} \right)^r \\
&\leq f_c(d,d) \cdot \sum_{r=0}^{d} \frac{c^r}{(r!)^2} \left(cd \ln \frac{m}{d} \right)^r \\
&\leq f_c(d,d) \cdot e^{2\sqrt{cd \ln \frac{m}{d}}} ,
\end{aligned}
$$

where the second inequality follows from Lemma 5.4 and the assumption that $c \geq 1$, and the last inequality follows from Lemma 4.3. Combined with Corollary 5.2 we get the claim of the lemma.

□

Lemma 5.6 *For all $d, m \geq 1$ and constants $c \geq 1$ and $\epsilon > 0$*

$$
\begin{aligned}
f_c(d,m) &\geq \mathrm{e}^{(2-o(1))\sqrt{d \max\{2, c \ln \frac{d+m}{2d}\}}} , \text{ if } d \leq m \leq \mathrm{e}^{d^{\frac{1-\epsilon}{c}}}, \\
f_c(d,m) &\geq \mathrm{e}^{(2-o(1))\sqrt{m \max\{2, c \ln \frac{d+m}{2m}\}}} , \text{ if } m < d \leq \mathrm{e}^{m^{\frac{1-\epsilon}{c}}}.
\end{aligned}
$$

In both cases

$$
f_c(d,m) = \min \left\{ 2^{\Omega(\sqrt{d \log \frac{m}{d}})}, 2^{\Omega(\sqrt{m \log \frac{d}{m}})} \right\} .
$$

Lemma 5.6 is proved by proving separate lower bounds for $f_c(d,d)$ and $f_c(d,m)$ where d and m are far apart. The lower bound for $f_c(d,d)$ uses Lemma 5.1, and the lower bound for $f_c(d,m)$ uses Lemma 5.4. Lemma 5.6 is proved in Appendix B.

6 The Improved-Random-Facet pivoting rule

We now get to the main result of this paper: description and analysis of a realizable pivoting rule that essentially matches the performance of the "pivoting rule" of the previous section. An important step in developing this new pivoting rule is a *change of perspective*. Although reaching the top vertex is our ultimate goal, we focus on the seemingly secondary goal of reaching a large number of facets. Reaching the top is now viewed as a (desired) side effect.

We start with an informal description of the improved pivoting rule which we call Improved-Random-Facet. Let (F, B_0) be the linear program to be solved, where B_0 is the initial basis, and let $d = |F \cap B_0|$ be the dimension. Let A be the set of active facets of (F, B_0) reached so far. For every $f \in A$, let B_f be a basis, with $f \in B_f$ and $v(B_f) \geq v(B_0)$, that defines the vertex reached on f. Initially $A = F \cap B_0$, $B_f = B_0$, for every $f \in F \cap B_0$, and $|A| = d$. Our goal is now to reach more active facets of (F, B_0) as quickly as possible. Let $c > 1$ be a parameter to be chosen later. We start in a manner similar to the one used by the basic Random-Facet pivoting rule. We choose a random active facet $f \in A$ and do a recursive call on $(F \setminus \{f\}, B_0)$. Each facet reached by this recursive call is also active of (F, B_0). Each facet reached during this recursive call is thus also added to A. (Note that each recursive call has its own local version of A.) Let B_1 be the basis returned by the recursive call on $(F \setminus \{f\}, B_0)$ and let $B_2 \leftarrow \textsc{Pivot}(F, B_1, f)$. If $B_1 = B_2$, then B_1 is the optimal solution of the problem (F, B_0) and we are done. If $B_1 \neq B_2$, then as before we do a second recursive call on (F, B_2). Facets reached during this recursive call are again active facets of (F, B_0) and are again added to A. We now get to the main point in which Improved-Random-Facet differs from Random-Facet. Whenever the size of A reaches the next power of c, rounded

Algorithm IRF(F, B_0, i)

if $F \cap B_0 = \emptyset$ **then return** (B_0, i)

$A_i \leftarrow F \cap B_0$
for $f \in A_i$ **do** $B_f \leftarrow B_0$

while *true* **do**
  $n_i \leftarrow \lceil\!\lceil |A_i| \rceil\!\rceil_c$
  $f \leftarrow$ RANDOM(A_i)
  $(B_1, j) \leftarrow$ IRF$(F \setminus \{f\}, B_f, i+1)$
  if $j < i$ **then return** (B_1, j)

  if $j = i$ **then continue**
  $B_2 \leftarrow$ PIVOT(F, B_1, f)
  if $B_1 = B_2$ **then return** (B_1, i)

  $f' \leftarrow B_2 \setminus B_1$; $B_{f'} \leftarrow B_2$
  for $j \leftarrow 0$ **to** i **do** $A_j \leftarrow A_j \cup \{f'\}$

  for $j \leftarrow 0$ **to** $i-1$ **do**
    if $|A_j| = n_j$ **then return** (B_2, j)
  if $|A_i| = n_i$ **then continue**
  $(B_3, j) \leftarrow$ IRF$(F, B_2, i+1)$
  if $j = i+1$ **then return** (B_3, i)

  if $j < i$ **then return** (B_3, j)

Figure 2: The simplex algorithm with the IMPROVED-RANDOM-FACET (IRF) pivoting rule.

up to the nearest integer, and this can happen either while the first recursive call on $(F \setminus \{f\}, B_0)$ is running, or while moving from B_1 to B_2, or while the second recursive call on (F, B_2) is running, we *abort* all running recursive calls, sample a new active facet from A, and start the whole process again. The intuition is that as A, the set of active facets already reached, gets larger and larger, the behavior of the algorithm becomes closer and closer to the behavior of IDEAL-RANDOM-FACET. We formalize this intuition below. The process of course ends when reaching the optimal vertex.

For $c > 1$, we let $\lceil\!\lceil n \rceil\!\rceil_c$ denote the smallest power of c, rounded up to the nearest integer, which is strictly larger than n. Similarly, we let $\lfloor\!\lfloor n \rfloor\!\rfloor_c$ denote the largest power of c, again rounded up, which is strictly smaller than n. We define $\lfloor\!\lfloor 1 \rfloor\!\rfloor_c$ to be 0.

A formal description of IMPROVED-RANDOM-FACET appears in Figure 2. For brevity, we shorten the name of the function to IRF. IRF takes a third argument i which is the *depth* of the recursion. A problem (F, B_0) is solved by calling IRF$(F, B_0, 0)$. At each stage, there is at most one active recursive call at each level. The instance at depth i has a set A_i of the active facets reached during the current recursive calls at depth i or larger. A recursive call at depth i adds the facets it

discovers to all sets A_j, for $j = 0, 1, \ldots, i$. A call at level i sets a target n_i. Whenever the size of A_i reaches n_i, all deeper recursive calls are aborted, a new target $n_i \leftarrow \lceil\!\lceil |A_i| \rceil\!\rceil_c$ is set, a new random facet from A_i is chosen, and a new iteration of the while loop at level i begins.

A call $\mathrm{IRF}(F, B_0, i)$ returns a pair (B, j), where B is the last basis reached and $j \leq i$ is an integer. If $j = i$, the returned basis B is optimal for (F, B_0). If $j < i$, the call was aborted as the target at level j was met. In that case, level j is the first level in which the target was met. A call $\mathrm{IRF}(F, B_0, 0)$ always returns an optimal solution, as there are no higher level targets to meet.

For $c > 1$, let $\bar{f}_{(c)}(d, n)$ be the function defined by the following recurrence relation:

$$\bar{f}_{(c)}(d,n) \;=\; \bar{f}_{(c)}(d, \lfloor\!\lfloor n \rfloor\!\rfloor_c) + \bar{f}_{(c)}(d-1, n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor_c} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor_c} (1 + \bar{f}_{(c)}(d, n-i)) \;,$$

for $0 < d < n$, and $\bar{f}_{(c)}(d, n) = 0$ for $d = 0$ or $n \leq d$.

Theorem 6.1 *For any $c > 1$, $\bar{f}_{(c)}(d, n+1)$ is an upper bound on the expected number of pivoting steps performed by the* Improved-Random-Facet *pivoting rule with parameter c, on a problem of dimension d with at most n active constraints with respect to the initial basis.*

Proof: We prove by induction that $\bar{f}_{(c)}(d, n)$ is an upper bound on the expected number of pivoting steps performed until reaching n active facets with respect to the initial basis, or until reaching the top vertex. If there are at most n active facets, the top must be reached before visiting $n+1$ active facets, and thus $\bar{f}_{(c)}(d, n+1)$ is an upper bound on the expected number of steps performed before reaching the top. We focus below on the top most level of the recursion, i.e., level 0, but it is easy to see that the argument holds for all levels of the recursion.

Let (F, B_0) be the problem to be solved. Let $d = |F \cap B_0|$, and let $n_0 = d$ and $n_j = \lceil\!\lceil n_{j-1} \rceil\!\rceil_c$, for $j > 0$. Let A_0 be the set of active facets gathered by the call $\mathrm{IRF}(F, B_0, 0)$. Whenever $|A_0|$ reaches n_j, for some $j \geq 0$, a new iteration of the while loop begins. By induction, the expected number of steps made until reaching $\lfloor\!\lfloor n \rfloor\!\rfloor_c < n$ active facets, or reaching the top, is at most $\bar{f}_{(c)}(d, \lfloor\!\lfloor n \rfloor\!\rfloor_c)$.

If the top is reached before collecting $\lfloor\!\lfloor n \rfloor\!\rfloor_c$ active facets, we are done. Otherwise, a new iteration of the while loop begins, with $|A_0| = \lfloor\!\lfloor n \rfloor\!\rfloor_c$. A new random facet $f \in A_0$ is sampled, and a recursive call on $(F \setminus \{f\}, B_f, 1)$ is made. By induction, the expected number of steps made by this recursive call until either reaching $n-1$ active facets, or reaching the top, is at most $\bar{f}_{(c)}(d-1, n-1)$. All active facets found by this recursive call are also active facets of (F, B_0) and are added by the recursive call not only to A_1 but also to A_0. Note that f is not an active facet of $(F \setminus \{f\}, B_f)$, as it does not belong to $F \setminus \{f\}$. Thus, if $n-1$ active facets were gathered by this first recursive call, then n active facets of (F, B_0) were reached, and we are done. (Note that this takes into account the possibility that some of the facets reached by the recursive call were already in A_0.)

If the recursive call $(F \setminus \{f\}, B_f, 1)$ finds the optimal basis B_1 of $(F \setminus \{f\}, B_f)$ before collecting $n-1$ active facets, the pivoting step $\mathrm{Pivot}(F, B_1, f)$ is attempted. If the step is unsuccessful, then B_1 is also an optimal basis of (F, B_0) and we are again done. If the pivoting step is successful, then the new basis B_2 satisfies $v(B_2) > v(B_1) = v(f)$. As in the previous sections, let $A_0 = \{f_1, f_2, \ldots, f_{\lfloor\!\lfloor n \rfloor\!\rfloor}\}$, where $\lfloor\!\lfloor n \rfloor\!\rfloor = \lfloor\!\lfloor n \rfloor\!\rfloor_c$, be the active facets arranged such that

$$v(f_1) \leq v(f_2) \;\leq\; \cdots \;\leq\; v(f_{\lfloor\!\lfloor n \rfloor\!\rfloor}) \;,$$

and suppose that $f = f_i$. The index i is uniformly distributed in $\{1, 2, \ldots, \lfloor\!\lfloor n \rfloor\!\rfloor\}$.

Next, a recursive call $(F, B_2, 1)$ is performed. Every basis B encountered during this recursive call satisfied $v(B) \geq v(B_2) > v(B_{f_i})$. In particular, the facets $f_1, f_2, \ldots, f_i$ cannot be encountered again

during this recursive call. Thus, by the time $n-i$ active facets of B_2 are reached by this recursive call, at least n active facets of B_0 have been collected. By induction, this happens after an expected number of at most $\bar{f}_{(c)}(d, n-i)$ steps. If the recursive call returns an optimal basis for (F, B_2), then this is also an optimal basis for (F, B_0) and we are again done. □

We next use a trick similar to a trick used by Gärtner [23]. Define two functions $\bar{g}_c(d,n)$ and $\bar{h}_c(n)$ by the recurrence relations:

$$\bar{g}_c(d,n) \;=\; \bar{g}_c(d-1,n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor_c} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor_c} (1 + \bar{g}_c(d,n-i))$$

for $d, n > 0$, and

$$\begin{aligned} \bar{g}_c(d,n) &= 0 \text{ , for } n \le d \text{ ,} \\ \bar{g}_c(0,n) &= 0 \text{ , for } n \ge 0 \text{ ,} \end{aligned}$$

and

$$\begin{aligned} \bar{h}_c(n) &= \bar{h}_c(n-1) + \bar{h}_c(\lfloor\!\lfloor n \rfloor\!\rfloor_c) \text{ , for } n > 0 \text{ ,} \\ \bar{h}_c(0) &= 1 \text{ .} \end{aligned}$$

Note that $\bar{g}_c(d,n)$ satisfies the same recurrence as $\bar{f}_{(c)}(d,n)$ with the first term omitted. Also note the similarity of $\bar{g}_c(d,n)$ to $\bar{f}_c(d,n)$.

Lemma 6.2 $\quad \bar{f}_{(c)}(d,n) \;\le\; \bar{g}_c(d,n)\,\bar{h}_c(n) \;\;,\;\; 0 \le d \le n \;.$

Proof: For brevity, we write $\bar{f}(d,n)$, $\bar{g}(d,n)$ and $\bar{h}(n)$ instead of $\bar{f}_{(c)}(d,n)$, $\bar{g}_c(d,n)$ and $\bar{h}_c(n)$, and $\lfloor\!\lfloor n \rfloor\!\rfloor$ instead of $\lfloor\!\lfloor n \rfloor\!\rfloor_c$. The claim easily follows by induction as

$$\begin{aligned} \bar{f}(d,n) &= \bar{f}(d, \lfloor\!\lfloor n \rfloor\!\rfloor) + \bar{f}(d-1,n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor} (1 + \bar{f}(d,n-i)) \\ &\le \bar{g}(d, \lfloor\!\lfloor n \rfloor\!\rfloor)\bar{h}(\lfloor\!\lfloor n \rfloor\!\rfloor) + \bar{g}(d-1,n-1)\bar{h}(n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor} (1 + \bar{g}(d,n-i))\bar{h}(n-i) \\ &\le \bar{g}(d,n)\bar{h}(\lfloor\!\lfloor n \rfloor\!\rfloor) + \bar{g}(d-1,n-1)\bar{h}(n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor} (1 + \bar{g}(d,n-i))\bar{h}(n-1) \\ &= \bar{g}(d,n)\bar{h}(\lfloor\!\lfloor n \rfloor\!\rfloor) + \left(\bar{g}(d-1,n-1) + \frac{1}{\lfloor\!\lfloor n \rfloor\!\rfloor} \sum_{i=1}^{\lfloor\!\lfloor n \rfloor\!\rfloor} (1 + \bar{g}(d,n-i)) \right) \bar{h}(n-1) \\ &= \bar{g}(d,n)\bar{h}(\lfloor\!\lfloor n \rfloor\!\rfloor) + \bar{g}(d,n)\bar{h}(n-1) \\ &= \bar{g}(d,n) \left(\bar{h}(\lfloor\!\lfloor n \rfloor\!\rfloor) + \bar{h}(n-1) \right) \\ &= \bar{g}(d,n)\bar{h}(n) \;. \end{aligned}$$

When moving from the second to third line we replaced $\bar{g}(d, \lfloor\!\lfloor n \rfloor\!\rfloor)$ by $\bar{g}(d,n)$ and $\bar{h}(n-i)$ by $\bar{h}(n-1)$, using the fact that $\bar{g}(d,n)$ and $\bar{h}(n)$ are monotonically non-decreasing in n. □

As $\lfloor\!\lfloor n \rfloor\!\rfloor_c \ge n/c$, the proof of Lemma 6.3 is immediate.

Lemma 6.3 $\bar{g}_c(d,n) \le \bar{f}_c(d,n)$.

Lemma 6.4 $\bar{h}_c(n) \le 4\mathrm{e}^{\frac{\ln^2(c^2 n)}{2\ln c}}$, *for* $n \ge 1$ *and* $c > 1$.

Proof: As $\lfloor\!\lfloor n' \rfloor\!\rfloor_c = \lfloor\!\lfloor n \rfloor\!\rfloor_c$ for every $\lfloor\!\lfloor n \rfloor\!\rfloor_c < n' \le n$, it follows easily by induction that

$$\bar{h}_c(n) = (n - \lfloor\!\lfloor n \rfloor\!\rfloor_c + 1)\,\bar{h}_c(\lfloor\!\lfloor n \rfloor\!\rfloor_c) .$$

As $\lfloor\!\lfloor \lceil c \rceil \rfloor\!\rfloor_c = 1$, we get that $\bar{h}_c(\lceil c \rceil) = \lceil c \rceil \bar{h}_c(1) = 2\lceil c \rceil \le 4c$.
Let $k = k_0 > 0$ and define the sequence $n_0 = \lceil c^{k_0} \rceil$, and $n_i = \lfloor\!\lfloor n_{i-1} \rfloor\!\rfloor_c = \lceil c^{k_i} \rceil$, for $i > 0$, until $n_r = 1$ and $k_r = 0$. For every i, we choose the minimal k_i satisfying the condition. (Note that we may have $\lceil c^{k-1} \rceil = \lceil c^k \rceil$, if $c < 2$ and k is small enough.) Note that $k_0 > k_1 > \cdots > k_r$ and that $k_0 = k$, $k_{r-1} = 1$, $k_r = 0$. Now, for $i < r - 1$ we have

$$\begin{aligned} \bar{h}_c(\lceil c^{k_i} \rceil) &= (\lceil c^{k_i} \rceil - \lceil c^{k_{i+1}} \rceil + 1)\,\bar{h}_c(\lceil c^{k_{i+1}} \rceil) \\ &\le c^{k_i} \bar{h}_c(\lceil c^{k_{i+1}} \rceil) . \end{aligned}$$

By induction we thus have

$$\begin{aligned} \bar{h}_c(\lceil c^k \rceil) &\le c^{k_0} c^{k_1} \cdots c^{k_{r-2}}\, \bar{h}_c(\lceil c^{k_{r-1}} \rceil) \\ &\le c^k c^{k-1} \cdots c^2\, \bar{h}_c(\lceil c \rceil) \\ &\le 4\, c^{k(k+1)/2} . \end{aligned}$$

Now, if $k = \lceil \log_c n \rceil$, then $n \le \lceil c^k \rceil$ and thus

$$\bar{h}_c(n) \le 4c^{k(k+1)/2} \le 4c^{(\log_c n + 2)^2/2} \le 4\mathrm{e}^{\frac{\ln^2(c^2 n)}{2\ln c}} .$$

□

Putting everything together, we get the main result of this paper:

Theorem 6.5 *Let* $0 \le d \le n$, *and let* $c = 1 + \epsilon$ *for some constant* $\epsilon > 0$. *If* $d \le n - d$ *then*

$$\bar{f}_{(c)}(d,n) \le \mathrm{e}^{(2+\epsilon)\sqrt{d \ln \frac{n-d}{d}} + O(\sqrt{d} + \log^2 n)} ,$$

and if $d \ge n - d$ *then*

$$\bar{f}_{(c)}(d,n) \le \mathrm{e}^{(2+\epsilon)\sqrt{(n-d) \ln \frac{d}{n-d}} + O(\sqrt{n-d} + \log^2 n)} .$$

It is possible to choose $c = 1 + o(1)$ in Theorem 6.5, but in this case the bounds for $\bar{h}_c(n)$ and $\bar{f}_c(d,n)$ need to be carefully balanced.

7 Concluding remarks

We obtained an improved version of the RANDOM-FACET pivoting rule of Kalai [33] and Matoušek, Sharir, and Welzl [46]. When $n = O(d)$, the expected running time is improved from $2^{O(\sqrt{d \log d})}$ to $2^{O(\sqrt{d})}$. The improved algorithm is a primal algorithm. We believe that it is essentially the fastest algorithm that can be obtained using the current ideas.

A conventional simplex algorithm moves from a vertex to a neighboring vertex, thus defining a *path* on the polytope. The improved pivoting rule is unconventional in the sense that it occasionally *jumps* back to a previously visited vertex. It would be interesting to know whether such jumps are necessary or whether they can be avoided.

Another interesting question is whether the lower bounds of Matoušek [44] and of Friedmann et al. [22] can also be made to work for our improved pivoting rule.

8 Acknowledgements

We would like to thank Gil Kalai, Bernd Gärtner, Günter Rote, and Tibor Szabó for many helpful discussions on the RANDOM-FACET algorithm and its variants.

References

[1] N. Amenta and G. Ziegler. Deformed products and maximal shadows of polytopes. In *Advances in Discrete and Computational Geometry*, pages 57–90, Providence, 1996. Amer. Math. Soc. Contemporary Mathematics 223.

[2] D. Avis and V. Chvátal. Notes on Bland's pivoting rule. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 24–34. Springer, 1978.

[3] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[4] H. Björklund and S. Vorobyov. Combinatorial structure and randomized subexponential algorithms for infinite games. *Theoretical Computer Science*, 349(3):347–360, 2005.

[5] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[6] R. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977.

[7] V. Chvátal. *Linear programming*. W. H. Freeman and Company, New York, 1983.

[8] K. Clarkson. Linear programming in $O(n \times 3^{d^2})$ time. *Information Processing Letters*, 22(1):21–24, 1986.

[9] K. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995.

[10] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[11] G. Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.

[12] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008.

[13] M. Dyer. On a multidimensional search technique and its application to the Euclidean one-centre problem. *SIAM J. Comput.*, 15(3):725–738, 1986.

[14] M. Dyer and A. Frieze. A randomized algorithm for fixed-dimensional linear programming. *Mathematical Programming*, 44(1-3):203–212, 1989.

[15] J. Fearnley. Exponential lower bounds for policy iteration. In *Proc. of 37th ICALP*, pages 551–562, 2010.

[16] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[17] O. Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Logical Methods in Computer Science*, 7(3), 2011.

[18] O. Friedmann. A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In *Proc. of 15th IPCO*, pages 192–206, 2011.

[19] O. Friedmann, T. D. Hansen, and U. Zwick. A subexponential lower bound for the random facet algorithm for parity games. In *Proc. of 22nd SODA*, pages 202–216, 2011.

[20] O. Friedmann, T. D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. of 43th STOC*, pages 283–292, 2011.

[21] O. Friedmann, T. D. Hansen, and U. Zwick. Errata for: A subexponential lower bound for the random facet algorithm for parity games. *arXiv*, arXiv:1410.7871 [cs.DS], 2014.

[22] O. Friedmann, T. D. Hansen, and U. Zwick. Random-Facet and Random-Bland require subexponential time even for shortest paths. *arXiv*, arXiv:1410.7530 [cs.DS], 2014.

[23] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24(5):1018–1035, 1995.

[24] B. Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures and Algorithms*, 20(3):353–381, 2002.

[25] B. Gärtner and V. Kaibel. Two new bounds for the random-edge simplex-algorithm. *SIAM J. Discrete Math.*, 21(1):178–190, 2007.

[26] B. Gärtner and I. Schurr. Linear programming and unique sink orientations. In *Proc. of 17th SODA*, pages 749–757, 2006.

[27] D. Goldfarb and W. Sit. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4):277 – 285, 1979.

[28] M. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACT News*, 26(2):96–104, 1995.

[29] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.

[30] T. D. Hansen, P. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1):1, 2013.

[31] T. D. Hansen, M. Paterson, and U. Zwick. Improved upper bounds for Random-Edge and Random-Jump on abstract cubes. In *Proc. of 25th SODA*, pages 874–881, 2014.

[32] R. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973.

[33] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. of 24th STOC*, pages 475–482, 1992.

[34] G. Kalai. Upper bounds for the diameter and height of graphs of convex polyhedra. *Discrete & Computational Geometry*, 8:363–372, 1992.

[35] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217–233, 1997.

[36] G. Kalai. Personal communication, 2012.

[37] G. Kalai and D. Kleitman. Quasi-polynomial bounds for the diameter of graphs and polyhedra. *Bull. Amer. Math. Soc.*, 26:315–316, 1992.

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

[39] J. Kelner and D. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proc. of 38th STOC*, pages 51–60, 2006.

[40] L. Khachiyan. A polynomial time algorithm in linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

[41] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

[42] V. Lifschitz and B. Pittel. The number of increasing subsequences of the random permutation. *Journal of Combinatorial Theory, Series A*, 31(1):1–20, 1981.

[43] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[44] J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures and Algorithms*, 5(4):591–608, 1994.

[45] J. Matoušek and B. Gärtner. *Understanding and using linear programming*. Springer, 2007.

[46] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[47] J. Matoušek and T. Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006.

[48] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

[49] M. Puterman. *Markov decision processes*. Wiley, 1994.

[50] A. Rasche. On Kalai's survey of linear programming and simple polytopes. Master's thesis, ETH Zürich, 2004.

[51] F. Santos. A counterexample to the Hirsch conjecture. *Annals of mathematics*, 176(1):383–412, 2012.

[52] F. Santos. Recent progress on the combinatorial diameter of polytopes and simplicial complexes. *TOP*, 21(3):426–460, 2013.

[53] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[54] I. Schurr and T. Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004.

[55] I. Schurr and T. Szabó. Jumping doesn't help in abstract cubes. In *Proc. of 9th IPCO*, pages 225–235, 2005.

[56] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6:423–434, 1991.

[57] D. Spielman and S. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[58] T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. of 42th FOCS*, pages 547–555, 2001.

[59] M. Todd. An improved Kalai-Kleitman bound for the diameter of a polyhedron. *arXiv*, arXiv:1402.3579 [math.OC], 2014.

[60] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.

A The recurrence relation of IDEAL-RANDOM-FACET

In this section we prove Lemma 5.3 by deriving interesting, alternative recurrence relations satisfied by the function $f_c(d,m)$, which bounds the number of expected steps made by the IDEAL-RANDOM-FACET"pivoting rule". Recall the definition of $f_c(d,m)$:

Definition A.1 *Let $f_c(d,m)$, for $d \geq 0$ and $m \geq 0$, be defined as*

$$\begin{aligned} f_c(d,m) &= f_c(d-1,m) + \frac{c}{d+m}\sum_{j=0}^{m-1} f_c(d,j) \quad , \quad d,m>0 \\ f_c(d,0) &= f_c(0,m) = 1 \end{aligned}$$

Definition A.2 *Let $F_c(d,m)$, for $d \geq 0$ and $m \geq 0$ be defined as*

$$F_c(d,m) = \sum_{j=0}^{m} f_c(d,j) \; .$$

Lemma A.3 $$F_c(d,m) = \left(1+\frac{c}{d+m}\right) F_c(d,m-1) + f_c(d-1,m) \quad , \quad d,m>0$$

Proof: For $d,m>0$, by the definition of $f_c(d,m)$ we have

$$F_c(d,m) - F_c(d,m-1) = f_c(d-1,m) + \frac{c}{d+m} F_c(d,m-1) \; ,$$

or equivalently

$$F_c(d,m) = \left(1+\frac{c}{d+m}\right) F_c(d,m-1) + f_c(d-1,m) \; .$$

□

Lemma A.4 $$F_c(d,m) = (d+m+1)\sum_{j=0}^{m} \frac{f_c(d-1,j)}{d+j+1} \prod_{t=j+1}^{m} \frac{d+t+c}{d+t+1} \quad , \quad \textit{for } d>0 \textit{ and } m \geq 0.$$

Proof: For every $d>0$ we prove the relation by induction on m. When $m=0$, both sides are 1, which is the basis of the induction. The induction step is:

$$\begin{aligned} F_c(d,m) &= \left(1+\frac{c}{d+m}\right) F_c(d,m-1) + f_c(d-1,m) \\ &= \frac{d+m+c}{d+m} \cdot (d+m) \sum_{j=0}^{m-1} \frac{f_c(d-1,j)}{d+j+1} \prod_{t=j+1}^{m-1} \frac{d+t+c}{d+t+1} + f_c(d-1,m) \\ &= (d+m+1) \sum_{j=0}^{m-1} \frac{f_c(d-1,j)}{d+j+1} \prod_{t=j+1}^{m} \frac{d+t+c}{d+t+1} + f_c(d-1,m) \\ &= (d+m+1) \sum_{j=0}^{m} \frac{f_c(d-1,j)}{d+j+1} \prod_{t=j+1}^{m} \frac{d+t+c}{d+t+1} \; . \end{aligned}$$

□

Note that for the idealized recurrence, $c=1$, we have $\prod_{t=j+1}^{m} \frac{d+t+c}{d+t+1} = 1$.

Lemma A.5 $$f_c(d,m) \;=\; f_c(d-1,m) + c\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\;,\;\; d,m>0\;.$$

Proof: The relation follows easily from the previous lemma:

$$\begin{aligned}
f_c(d,m) &= F_c(d,m) - F_c(d,m-1)\\
&= (d+m+1)\sum_{j=0}^{m}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m}\frac{d+t+c}{d+t+1} -\\
&\quad (d+m)\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\\
&= f_c(d-1,m) + (d+m+c)\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1} -\\
&\quad (d+m)\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\\
&= f_c(d-1,m) + c\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\;.
\end{aligned}$$

□

Lemma A.6 *For $d \geq 0$ and $m \geq 0$ we have:*

$$f_c(d,m) \;=\; 1 + \sum_{i=0}^{d-1}\sum_{j=0}^{m-1}\frac{c\,f_c(i,j)}{i+j+2}\prod_{t=j+1}^{m-1}\frac{i+t+1+c}{i+t+2}$$

Proof: For $d=0$ or $m=0$ the sum is vacuous. For $d=1$, this is exactly the claim of Lemma A.5. For $d>1$ we get by induction, again using Lemma A.5:

$$\begin{aligned}
f_c(d,m) &= f_c(d-1,m) + c\sum_{j=0}^{m-1}\frac{f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\\
&= 1 + \sum_{i=0}^{d-2}\sum_{j=0}^{m-1}\frac{c\,f_c(i,j)}{i+j+2}\prod_{t=j+1}^{m-1}\frac{i+t+1+c}{i+t+2} + \sum_{j=0}^{m-1}\frac{c\,f_c(d-1,j)}{d+j+1}\prod_{t=j+1}^{m-1}\frac{d+t+c}{d+t+1}\\
&= 1 + \sum_{i=0}^{d-1}\sum_{j=0}^{m-1}\frac{c\,f_c(i,j)}{i+j+2}\prod_{t=j+1}^{m-1}\frac{i+t+1+c}{i+t+2}\;.
\end{aligned}$$

□

For $c=1$ the identity of Lemma A.6 simplifies to:

$$f_1(d,m) \;=\; 1 + \sum_{i=0}^{d-1}\sum_{j=0}^{m-1}\frac{f_1(i,j)}{i+j+2}\;.$$

As an interesting corollary, we thus get that $f_c(d,m)$ is *symmetric* for $c=1$, which is not apparent from the original definition.

Corollary A.7 *For every $d \geq 0$ and $m \geq 0$, we have $f_1(d,m) = f_1(m,d)$.*

B Lower bound for the IMPROVED-RANDOM-FACET recurrence

In this section we prove Lemma 5.6 by proving and combining two separate lower bounds.

Lemma B.1 *For all $d, m \geq 1$, $c \geq 1$, and $\epsilon > 0$ satisfying $d, m \leq \mathrm{e}^{(\min\{d,m\})^{1-2\epsilon}/c}$, we have:*

$$f_c(d,m) \;\geq\; \frac{\mathrm{e}^{2\sqrt{\frac{c\min\{d,m\}}{1+2d^{-\epsilon}}\ln\frac{d+m}{2\min\{d,m\}}}}}{\mathrm{e}^2 d^3}\,.$$

Proof: The lower bound is obtained using the "closed-form" of Lemma 5.4. Since $c \geq 1$ we have:

$$\begin{aligned} f_c(d,m) \;&=\; \sum_{\substack{1\leq d_1<\cdots<d_k\leq d\\ 1\leq m_1<\cdots<m_{k+1}=m+1}} \prod_{i=1}^{k} \frac{c}{d_i+m_i} \prod_{t=m_i+1}^{m_{i+1}-1}\left(1+\frac{c-1}{d_i+t}\right) \\ &\geq\; \sum_{\substack{1\leq d_1<\cdots<d_k\leq d\\ 1\leq m_1<\cdots<m_k\leq m}} \prod_{i=1}^{k} \frac{c}{d_i+m_i}\,. \end{aligned}$$

Note that this sum is symmetric in d and m. For the remainder of the proof we therefore assume that $d \leq m$. By using that $d_i \leq d$ for all $i \leq k$, we get:

$$\begin{aligned} \sum_{\substack{1\leq d_1<\cdots<d_k\leq d\\ 1\leq m_1<\cdots<m_k\leq m}} \prod_{i=1}^{k} \frac{c}{d_i+m_i} \;&\geq\; \sum_{\substack{1\leq d_1<\cdots<d_k\leq d\\ 1\leq m_1<\cdots<m_k\leq m}} \prod_{i=1}^{k} \frac{c}{d+m_i} \\ &=\; \sum_{k=0}^{d}\binom{d}{k} \sum_{1\leq m_1<\cdots<m_k\leq m} \prod_{i=1}^{k} \frac{c}{d+m_i}\,. \end{aligned}$$

To continue manipulating the lower bound, we observe that every sequence $d \leq m'_1 \leq \cdots \leq m'_k \leq m$ maps uniquely to a sequence $1 \leq m_1 < \cdots < m_k \leq m$ where $m_i = m'_i - k + i$. Since $m_i \leq m'_i$ we get (see more explanations below):

$$\begin{aligned} \sum_{k=0}^{d}\binom{d}{k} \sum_{1\leq m_1<\cdots<m_k\leq m} \prod_{i=1}^{k} \frac{c}{d+m_i} \;&\geq\; \sum_{k=0}^{d}\binom{d}{k} \sum_{d\leq m'_1\leq\cdots\leq m'_k\leq m} \prod_{i=1}^{k} \frac{c}{d+m'_i} \\ &\geq\; \sum_{k=0}^{d}\frac{1}{k!}\binom{d}{k} \sum_{i\in[k]:\, d\leq m'_i\leq m} \prod_{i=1}^{k} \frac{c}{d+m'_i} \\ &=\; \sum_{k=0}^{d}\frac{1}{k!}\binom{d}{k} \left(\sum_{i=2d}^{d+m}\frac{c}{i}\right)^k \\ &\geq\; \sum_{k=0}^{d}\frac{1}{k!}\binom{d}{k} \left(c\ln\frac{d+m}{2d}\right)^k \\ &\geq\; \frac{1}{r!}\binom{d}{r} \left(c\ln\frac{d+m}{2d}\right)^r\,, \end{aligned}$$

where $r \leq d^{1-\epsilon}$ will be specified later. To prove the second inequality, observe that the sum on the left-hand-side is over ordered sequences, while the sum on the right-hand-side is over unordered sequences. The inequality follows from the fact that there are $k!$ ways of permuting an ordered

sequence of distinct elements, and that there are less than $k!$ permutations when the elements are not distinct. The third inequality follows from $\sum_{i=2d}^{d+m} \frac{1}{i} \geq \int_{2d}^{d+m} \frac{1}{x} dx = \ln \frac{d+m}{2d}$.
Since $1 - x \geq \mathrm{e}^{-2x}$ for $x \leq \frac{1}{2}$ it follows that:

$$\frac{\binom{d}{r}}{\frac{d^r}{r!}} \;=\; \frac{d!}{d^r(d-r)!} \;=\; \prod_{i=1}^{r-1}\left(1 - \frac{i}{d}\right) \;\geq\; \prod_{i=1}^{r-1} \mathrm{e}^{\frac{-2i}{d}} \;=\; \mathrm{e}^{\frac{-2}{d}\sum_{i=1}^{r-1} i} \;\geq\; \mathrm{e}^{\frac{-r^2}{d}} \;,$$

which leads to the conclusion:

$$\begin{aligned} f_c(d,m) \;&\geq\; \frac{1}{(r!)^2}\left(\frac{cd}{\mathrm{e}^{r/d}} \ln \frac{d+m}{2d}\right)^r \\ &\geq\; \frac{1}{(r!)^2}\left(\frac{cd}{1+2r/d} \ln \frac{d+m}{2d}\right)^r \\ &\geq\; \frac{1}{(r!)^2}\left(\frac{cd}{1+2d^{-\epsilon}} \ln \frac{d+m}{2d}\right)^r \;, \end{aligned}$$

where the second inequality uses that $\mathrm{e}^x \leq 1 + 2x$ for $0 \leq x \leq 1$, and the third inequality uses the assumption that $r \leq d^{1-\epsilon}$.
We next let $r = \left\lceil \sqrt{\frac{cd}{1+2d^{-\epsilon}} \ln \frac{d+m}{2d}} \right\rceil$. Note that

$$\sqrt{\frac{cd}{1+2d^{-\epsilon}} \ln \frac{d+m}{2d}} \;\leq\; d^{1-\epsilon} \qquad\Longleftrightarrow\qquad m \;\leq\; 2d\mathrm{e}^{(1+2d^{-\epsilon})d^{1-2\epsilon}/c} - d \;,$$

and thus $r \leq d^{1-\epsilon}$ due to the assumption that $m \leq \mathrm{e}^{d^{1-2\epsilon}/c}$. Using that $r! \leq \mathrm{e}\sqrt{r}(r/\mathrm{e})^r$ we then get:

$$\frac{1}{(r!)^2}\left(\frac{cd}{1+2d^{-\epsilon}} \ln \frac{d+m}{2d}\right)^r \;\geq\; \frac{(r-1)^{2r}}{(r!)^2} \;\geq\; \frac{1}{\mathrm{e}^2 r^3}\,\mathrm{e}^{2r} \;\geq\; \frac{\mathrm{e}^{2\sqrt{\frac{cd}{1+2d^{-\epsilon}} \ln \frac{d+m}{2d}}}}{\mathrm{e}^2 d^3} \;.$$

□

When m and d are close, Lemma B.1 does not provide a good lower bound. In this case we instead lower bound $f_c(d,m)$ by $f_1(\delta,\delta)$, where $\delta = \min\{d,m\}$. Lemma 5.1 shows that:

$$f_1(2\delta) \;=\; \sum_{i=0}^{2\delta} f_1(i, 2\delta - i) \;.$$

We next show that the largest term in this sum is for $i = \delta$, from which it follows that $f_1(2\delta) \leq (2\delta+1)\, f_1(\delta,\delta)$. Recall that Lemma 4.5 shows that $f_c(k) \geq \frac{\mathrm{e}^{2\sqrt{ck}}}{\mathrm{e}^{3+c}(1+\sqrt{ck})}$ for every $k \geq 0$ and $1 \leq c \leq k$. Combining these observations gives:

$$f_c(d,m) \;\geq\; f_1(\delta,\delta) \;\geq\; \frac{f_1(2\delta)}{2\delta+1} \;\geq\; \frac{\mathrm{e}^{2\sqrt{2\delta}}}{\mathrm{e}^4(1+\sqrt{2\delta})^3} \;.$$

Lemma B.2 *$f_1(r, n-r) \geq f_1(s, n-s)$ for $|r - n/2| \leq |s - n/2|$, and $0 \leq r, s \leq n$.*

Proof: We prove the lemma by induction on n. For $n = 0$ the statement is true since $r = s = 0$. For the remainder of the proof we assume that $n > 0$. Since Corollary A.7 shows that $f_1(d,m) = f_1(m,d)$, we may assume that $n/2 \leq r \leq s$. Observe also that it suffices to prove the lemma for $s = r + 1$ since the statement then follows by induction.

From Lemma A.6 we get that:

$$\begin{aligned} f_1(r, n-r) &= 1 + \sum_{i=0}^{r-1} \sum_{j=0}^{n-r-1} \frac{f_1(i,j)}{i+j+2} \\ &= \left(1 + \sum_{i=0}^{r-1} \sum_{j=0}^{n-r-2} \frac{f_1(i,j)}{i+j+2}\right) + \sum_{i=0}^{r-1} \frac{f_1(i, n-r-1)}{i+n-r+1} \\ &\geq \left(1 + \sum_{i=0}^{r-1} \sum_{j=0}^{n-r-2} \frac{f_1(i,j)}{i+j+2}\right) + \sum_{i=2r-n+1}^{r-1} \frac{f_1(i, n-r-1)}{i+n-r+1} \\ &= \left(1 + \sum_{i=0}^{r-1} \sum_{j=0}^{n-r-2} \frac{f_1(i,j)}{i+j+2}\right) + \sum_{i=0}^{n-r-2} \frac{f_1(i+2r-n+1, n-r-1)}{i+r+2} \end{aligned}$$

and

$$\begin{aligned} f_1(r+1, n-r-1) &= 1 + \sum_{i=0}^{r} \sum_{j=0}^{n-r-2} \frac{f_1(i,j)}{i+j+2} \\ &= \left(1 + \sum_{i=0}^{r-1} \sum_{j=0}^{n-r-2} \frac{f_1(i,j)}{i+j+2}\right) + \sum_{j=0}^{n-r-2} \frac{f_1(r,j)}{r+j+2} \ . \end{aligned}$$

To complete the proof, we show that

$$\sum_{i=0}^{n-r-2} \frac{f_1(i+2r-n+1, n-r-1)}{i+r+2} \ \geq \ \sum_{i=0}^{n-r-2} \frac{f_1(r,i)}{r+i+2} \ .$$

We use the induction hypothesis to show that $f_1(i+2r-n+1, n-r-1) \geq f(r,i)$ for all $0 \leq i \leq n-r-2$. Observe that $i+2r-n+1+n-r-1 = i+r < n$. To use the induction hypothesis we show that $|i+2r-n+1-(i+r)/2| \leq |r-(i+r)/2|$.

Note that

$$i+2r-n+1-(i+r)/2 \ \geq \ 0 \qquad \Longleftrightarrow \qquad i \ \geq \ 2n-3r-2 \ ,$$

and

$$r-(i+r)/2 \ \geq \ 0 \qquad \Longleftrightarrow \qquad r \ \geq \ i \ .$$

Since $i \leq n-r-2$ and $n/2 \leq r$ we always have $i < r$. For $2n-3r-2 \leq i \leq n-r-2$ we thus get:

$$\begin{aligned} |i+2r-n+1-(r+i)/2| \ &= \ \frac{i}{2} + \frac{3r}{2} - n + 1 \ \leq \ \frac{r-i}{2} \ = \ |r-(r+i)/2| \\ &\Longleftrightarrow \qquad i \ \leq \ n-r-1 \ . \end{aligned}$$

For $0 \leq i < 2n-3r-2$ we have:

$$\begin{aligned} |i+2r-n+1-(r+i)/2| \ &= \ n - \frac{i}{2} - \frac{3r}{2} - 1 \ \leq \ \frac{r-i}{2} \ = \ |r-(r+i)/2| \\ &\Longleftrightarrow \qquad n \ \leq \ 2r+1 \ . \end{aligned}$$

□

Corollary B.3 *For all $d, m \geq 0$ and $c \geq 1$,*

$$f_c(d,m) \;\geq\; \frac{\mathrm{e}^2\sqrt{2\min\{d,m\}}}{\mathrm{e}^4(1+\sqrt{2\min\{d,m\}})^3}\ .$$

Combining Lemma B.1 and Corollary B.3 we get:

Lemma B.4 *For all $d, m \geq 1$, $c \geq 1$, and $\epsilon > 0$ satisfying $d, m \leq \mathrm{e}^{(\min\{d,m\})^{1-2\epsilon}/c}$, we have:*

$$f_c(d,m) \;\geq\; \frac{\mathrm{e}^2\sqrt{\min\{d,m\}\max\{2,\frac{c}{1+2d^{-\epsilon}}\ln\frac{d+m}{2\min\{d,m\}}\}}}{\mathrm{e}^4(1+2\min\{d,m\})^3}\ .$$