Seminar: Selected Topics in Logic and Verification

A SUMMARY ON: A PIVOTING ALGORITHM FOR CONVEX HULLS AND VERTEX ENUMERATION OF ARRANGEMENTS AND POLYHEDRA

September 30, 2020

Tilman Hinnerichs
Matrikelnummer: 4643427
Technische Universität Dresden

Tutor: Dr. Florian Funke

Summer semester 2020

Abstract

Contents

1 Introduction

This summary outlines the approach from Avis and Fukuda to the Vertex Enumeration problem mollified with various references to basic concepts such as the Simplex algorithm.

The Vertex Enumeration Problem (VEP) is one of the most fundamental problems in computational geometry and has applications in fields such as computational biology and quantum physics.

The paper „A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra" was written by David Avis and Komei Fukuda. It was published in 1992 in the journal „Discrete & Computational Geometry" and is regarded as one of the most cited papers in this very field with a number of 751 citations (as of 26.09.2020, according to Google Scholar). As a non-book, non-survey paper publication in such a specific field of research, this can be considered a fairly high amount of citations, emphasizing its importance.

We will thus follow a similar outline by first introducing polyhedra and arrangements, and the VEP on those. We then observe linear optimization problems and how they can be solved by the Simplex algorithm.

2 Polyhedra and Arrangements

In this first section we will introduce the needed notations and basics for the Vertex Enumeration Problem (VEP). At first, we will have to clarify the terms and concepts of the *vertex* and *polyhedra*.

Definition 1. *(Polyhedron)*
Given a matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$.

A convex polyhedron G is defined as

$$G = \{x \in \mathbb{R}^n : Ax + b \geq 0\} \tag{1}$$

Remark. *In general, this is the definition of a polytope, as polyhedra are the 3-dimensional special case of a polytope. As Avis and Fukuda use these terms interchangeably, we will do so as well in the following.*

This definition states that the set of points is constrained by a set of m linear inequalities with n that are basically hyperplanes in the n dimensional space. In general, we assume that $G \neq \emptyset$, if not stated otherwise. In the following we will abbreviate *convex polyhedra* simply as *polyhedra.*

An *arrangement of hyperplanes* or *arrangement* for short, is a decomposition of the given underlying space using a set of hyperplanes/ linear inequalities. As both concepts are fairly similar, we will not explicitly list *arrangements* in every definition, as every following statement holds for both polyhedra and arrangements. If that is not the case, arrangements will be named separately.

As the convex property is crucial for this approach to work, we will define it using convex combinations.

Definition 2. *(Convex combination)*

Let $x_1, x_2, \ldots, x_n$ be a finite set of points. A convex combination is a point of the form

$$y = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

while $a_1, \ldots, a_n \in \mathbb{R}^+$*, and*

$$a_1 + \cdots + a_n = 1$$

A convex combination is a linear combination.

Based on this definition we can further describe the constraints to our polyhedra and arrangements.

Definition 3. *A set is called convex if it contains all convex combinations of its points.*

As a polytope is in fact a set of points, too, this also applies to the polyhedra and arrangements.

Definition 4. *For a given set of points, the convex hull is the set set of all convex combinations of these points.*

In order to fully understand the last part of the title we additionally have to clarify the term *vertex*, which we do by the following definition.

Definition 5. *(Vertex)*
A point $v \in G$ *is called a vertex of any set* G ***iff*** *there are no two other points* $a, b \in G$ *such that*

$$v = \lambda_1 a + \lambda_2 b \tag{2}$$

which is also called a convex combination of a, b*.*

In order to make this definition a little more specific with regard to the definition of a polahedron, we will introduce our first corollary.

Corollary 1. *[1]*
A point $v \in G$ *is a vertex of* G ***iff*** *it is the unique solution to a subset of* m *inequalities solved as equations.*

3 The Vertex Enumeration Problem

3.1 Definition and general remarks

We are now ready to introduce the VEP itself with the following definition.

Definition 6. *(Vertex Enumeration Problem)*
For a given polytope/polyhedron or hyperplane arrangement G *determine all vertices of the object given its formal representation.*

Theorem 1. *(Complexity of the VEP)*
The VEP is NP-hard for unbounded polyhedra.[2]

The VEP is dual to the *Facet Enumeration problem*(FEP), that is finding all facets of a convex hull for a given set of points. The solution to the dual problem also yields the solution to the primal problem, too. We will introduce the term of duality in the following chapter.

While these recent definitions lack the ability to intrinsically motivate the applicability of this very problem, we will introduce a fairly simple example, which was taken and simplified from Avis' lecture notes.

Example 1. *(All meals for 1 Euro)*
Lets assume you are on a diet and additionally quite poor or minimalistic. Thus, you would like to plan your future meals, such that they only contain 3 different ingredients. Additionally, we would love to have sufficient nutrition intake, where we will only consider energy, protein and calcium intake.
Our decision is also constrained by the fact that we will only have 1 Euro per day, and can also endure just up to a certain amount of each ingredient, in order not just eat dry oatmeal all day

long.
The values describing our problem are given in the following table:

Var	Food type	Energy (kcal/100g)	Protein (g)	Calcium (mg)	Price (Euro cents)	Maximum
x_1	*Oatmeal*[1]	*250*	*15*	*40*	*10*	*500g*
x_2	*Milk*[2]	*80*	*4*	*130*	*4*	*1000g*
x_3	*DD Stollen*[3]	*340*	*7*	*30*	*20*	*200g*
	Min. daily	***2000***	***55***	***800***		

The variables x_1, x_2, x_3 each denote the amount of each ingredient in 100g.

By solving the VEP, we yield all vertices of the corresponding polyhedron, that is all points that cannot be written as convex combinations of two other points. Inverting this very statement, all other points within P are convex combinations of at least two other points. More specifically we are able to rewrite each point within P as convex combinations of vertices of P. Thus, if we find all vertices of the given arrangement of linear inequalities, we can produce infinitely many food combinations by randomly choosing a convex combinations of the given vertices. However, we still have to find these vertices.

3.2 Types of approaches

In history there have been various approaches to the VEP, that can be classified into two categories. First, there are the so called *Pivot based* methods, where this very method can be placed, too. These methods rely on traversing along the vertices using so called SIMPLEX tableaus, which we will introduce in the following chapter. These approaches are mostly based on solving corresponding linear optimization problems.

The other method is called *Fourier-Motzkin* or *double description* method[3]. These methods yield the vertices by successively adding hyperplanes and keeping track of the remaining possible vertices. The double description method in fact solves the dual problem of FEP.

While this classification was made in 1992 within this very publication, there haven't been exceptions to this ever since. We will discuss this in more detail in the last chapter.

4 Linear Programs

4.1 What is a Linear Optimization Problem?

While linear optimization problems are in general not necessary in order to solve the VEP with a pivot based method, it was utilized for this very approach. Additionally it gives us the chance to discuss and introduce some definitions and theorems that underline its simplicity and effectiveness. In the following we will use the terms *linear program*(LP) and *linear optimization problem* interchangeably.

Definition 7. *Let $G \subseteq \mathbb{R}^n$ be a polyhedron and $c \in \mathbb{R}^n$. Then a problem of the form*

$$z = f(x) = c^T \cdot x \to min \qquad with\ x \in G$$

is called a linear *optimization problem.*

Hereby P describes the feasible space as a set of n-dimensional vectors, while the objective function is denoted with $f : P \to \mathbb{R}$, which maps the vectors into the real numbers. Additionally, we can easily see the type of optimization, here minimization, while also a maximization problem is possible. Without loss of generality we will only consider minimization problems here. The linearity of the problem is derived from the linearity of the objective function itself.

Furthermore, we are able to bring in a statement about the *standard form* of linear programs.

Corollary 2. *All finite dim. linear optimization problems can be written in the following form:*

$$z = f(x) = c^T \cdot x \to min \qquad with\ x \in G := \{x \in \mathbb{R}^n : Ax \geq b, x \geq 0\} \tag{3}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \geq 0$. Additionally, let $rank(A) = m$ and $m < n$.
We call this form the standard form.

The finite dimensionality is expressed with regard to the amount of variables, and thus the width of both objective function and coefficient matrix.

Hereby, the notation $x \geq 0$ with $x \in \mathbb{R}$ describes the expression $x_j \geq 0$ for every $j = 1, 2, \ldots, n$.
Please note that we naturally do not have any objective function in the original VEP formulation. For now, lets assume that we do have a any linear objective function f given. We will discuss this issue in further detail later on.
After this quick introduction into the field of linear programs, we will now explore their solutions and the existence of those. We will use the following terminology.

Definition 8. *Let $\bar{X} \in \mathbb{R}^n$ be a solution to a linear program in standard form 3.*

(1) If $\bar{x}$ satisfies the linear equality system defined by $A\bar{x} \geq b, \bar{x} \geq 0$ (see 3), we call $x\bar{x}$ as a feasible solution. *Otherwise we will call $\bar{x}$ a infeasible solution.*

(2) $\bar{x}$ is optimal, *if for its objective value $\bar{z} = f(\bar{x})$ with objective function f the following applies:*

$$\bar{z} = f(\bar{x}) \leq z = f(x) \text{ for all } x \in G$$

If a linear program has a feasible solution but no optimal one, we will call t his LP unbounded.

4.2 Formulating LPs over Arrangements

Constructing an LP for a given hyperplane arrangement is fairly hard, as it is given by equations $y = Ax + b$ rather than inequalities. In order to reduce dimensionality, we choose m linearly independent equations and solve for x. After adding a random objective function, we interpret the result as a Simplex tableau. As its quite easy to check whether the normal vectors are parallel, we assume that the arrangement has at least on vertex.

4.3 Duality of LP

Linear programs can be reformulated, by basically „swapping“ the coefficient matrix, therefor introducing new variables and reformulating the linear inequalities. We will not go into further details here, as we are just interested in the yielded results. At first glance, the original or *primal* inequality system and its dual formulation have not that much in common, but in fact they share and mutually propagate many interesting properties.

The first property to be mentioned here is their reciprocal bound, as the weak duality theorem states.

Theorem 2. *(Weak Duality Theorem)*
Given

- *a primal LP* (P) *defined by* $c^T \cdot x \to \min$ *with* $Ax \geq b, x \geq 0$*, and*
- *its dual LP* (D) *defined by* $b^T \cdot u \to \max$ *with* $A^T u \leq c, u \geq 0$.

Let x *be feasible for* (P) *and* u *feasible for* (D)*. Then*

$$b^T \cdot u \leq c^T \cdot x$$

holds.

This law states, that any feasible solution of (P) will always be larger or equal to any feasible solution of (D).
We can proof an even stricter bound for both LPs.

Theorem 3. *(Strong Duality Theorem)*
The LP (P) *is solvable iff its dual LP* (D) *is solvable. For an optimal solution* $\bar{x}$ *of* (P) *and* $\bar{u}$ *of* (D)

$$b^T \cdot \bar{u} = c^T \cdot \bar{x}$$

holds, i.e. equality of the optimal solutions.

These theorems will help us better understand the eventual algorithm and its feasibility.

5 Simplex algorithm

5.1 Calculating a first Simplex tableau from the Standard Form

Before we are able to start applying the Simplex algorithm to a LP we need to transform it to a viable form for the procedure. Thus, we will transform a LP in the standard form into a so called *Simplex tableau*.

Definition 9. *(Simplex tableau)*
A LP of the form

$$z = f'(x') = c'^T \cdot x' \to min \qquad \text{with } x' \in G' := \{x' \in \mathbb{R}^{n+m} : A'x' = b, x' \geq 0\} \tag{4}$$

$$\text{with} \quad \begin{array}{l} A' \in \mathbb{R}^{m \times (m+n)} \\ b \in \mathbb{R}^m \end{array}$$

is called a Simplex tableau.

In contrast to the standard form the feasible space is now restricted by linear equations rather than inequalities. Additionally the dimensionality of x' has changed to $m + n$. These rather absurd looking changes are motivated by introducing so called *slack variables* that are added to the smaller side of the inequality. We thus introduce a new variable $s \geq 0$ for each inequality that describes the difference of both sides. For example we can transform the inequality $x \leq y$ to the equation $x + s = y$. This new equation is equivalent in meaning to its predecessor inequality.

Example 2. *As our introductory example is way too large to efficiently show the power of the Simplex algorithm, we will reduce this problem to two variables and two new inequalities as listed below. We also add a random objective function. This system is already given in standard form.*

$$\begin{array}{rcrcrcl} -z & = & -30a & - & 45b & \to & \boldsymbol{min} \\ \\ \text{with} & & 4a & + & 3b & \leq & 100 \\ & & a & + & 2b & \leq & 50 \\ & & a & , & b & \geq & 0 \end{array} \tag{5}$$

Which we will now transform to a Simplex tableau.

$$\begin{array}{rccccccccc} -z & = & -30x_1 & - & 45x_2 & \rightarrow & min & & \\ with & & -4x_1 & - & 3x_2 & = & -100 & + & s_1 \\ & & -x_1 & - & 2x_2 & = & -50 & + & s_2 \\ & & x_1 & , & x_2 & \geq & 0 & , & s_1, s_2 \geq 0 \end{array} \tag{6}$$

From now on we will also rename s_1, s_2 to x_3, x_4.

$$\begin{array}{rccccccccc} -z & = & -30x_1 & - & 45x_2 & \rightarrow & min & & \\ with & & -4x_1 & - & 3x_2 & = & -100 & + & x_3 \\ & & -x_1 & - & 2x_2 & = & -50 & + & x_4 \\ & & x_1 & , & x_2 & \geq & 0 & , & x_3, x_4 \geq 0 \end{array} \tag{7}$$

At this stage we will introduce the terms *basic solution* and *basic variables.* Let $I = \{1, \ldots, m+n\}$ be the index set denoting the given indices in the Simplex tableau. As $rank(A) = m$ holds, $rank(A') = m$ follows, and therefor there has to be subset $B \subset I$ with $|B| = m$, such that the columns $A^i, i \in B$ with A^i denoting the i-th column, are linearly independent. We hence denote B as the *basis index set* describing the *basic variables* x_B, and the *non-basis index set* $N = I \backslash B$ describing the *non-basic variables* x_N.

A basic solution is defined with respect to the given equation system as a unique solution to a subset of m inequalities solved as equations. This definition is similar to the characterization of corollary 1. In fact, this corollary is derived from the relation of basic solutions and vertex, such that each vertex has at least one basic solution. Thus, we will now use these terms interchangeably.

Definition 10. *A basic solution v of polytope G is called* feasible*, iff $v \in G$.*

5.2 The Simplex Algorithm

The Simplex algorithm can be divided into two phases.

Algorithm 1 The SIMPLEX algorithm

1: Determine a first feasible solution (vertex)
2: Determine the optimal solution

The algorithm starts by determining a first feasible solution, which is a vertex of our feasible polytope and thus a basic solution. It then iterates over all vertices in the feasible direction of the objective function.

Lemma 1. *If $G' \neq \emptyset$, then G' has at least one vertex and at most finitely many vertices.*

Thus, we should be able to find a first feasible basic solution.

This method is in fact only the *primal* Simplex algorithm, which can be extended to the *dual* Simplex algorithm. The latter is a procedure to solve the dual version of the primal LP. We will call a subset of indices B *dual feasible* if its feasible for the dual LP, and *dual infeasible* otherwise.

5.2.1 Phase 1: Determine a first feasible solution (vertex)

Finding a first can be both really easy and hard. In general a first basic solution can be obtained by

$$x' \longleftrightarrow \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} p \\ 0 \end{pmatrix} = \begin{pmatrix} A_B^{-1} \cdot b \\ 0 \end{pmatrix}$$

thus setting $x_N = 0$ and $x_B = p$, if that solution is feasible. However, if that is not the case, we can formulate a aid LP, given by

$$\begin{aligned} & h = e^T \cdot y \to min \\ \text{with} \quad & y + Ax = b, \\ & x, y \geq 0, \\ & x \in \mathbb{R}^{(m+n)}, y \in \mathbb{R}^m \\ & e = (1, \dots, 1)^T \in \mathbb{R}^m \end{aligned} \tag{8}$$

through which we can always find a feasible basic solution if there exists one.
In the case of our example problem there exists a basic solution

$$x' = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 100 \\ 50 \end{pmatrix}$$

Additionally, we can rewrite our equation system in the following form

ST_0	x_1	x_2	1
$x_3 =$	-4	-3	100
$x_4 =$	-1	-2	50
$-z =$	-30	-45	0

or in general for all LPs in form of a Simplex tableau

$$\begin{array}{r|c|c} ST_0 & x_N & 1 \\ \hline x_B = & P & p \\ & & \\ \hline z = & q^T & q_0 \end{array} \tag{9}$$

as we rename all variables for the upcoming Simplex algorithm steps. We will always have the non-basic variables on top and the basic variables on the left. As we found our first feasible solution we are ready for the second phase.

5.2.2 Phase 2: Determine the optimal solution

For each step of the algorithm we first determine a so called *Pivot element* $P_{\sigma\tau}$ by finding the column $\tau \in N$ chosen randomly, and the row $\sigma \in B$, which is determined by

$$-\frac{p_\sigma}{P_{\sigma\tau}} := \min \left\{ -\frac{p_i}{P_{i\tau}} \;\middle|\; -\frac{p_i}{P_{i\tau}} < 0, i \in B \right\} \tag{10}$$

Hereby, we choose the row i with the minimal $-\frac{p_i}{P_{i\tau}}$ and save this index to σ. The minimal quotient determines the maximum step size along the feasible direction. If $-\frac{p_i}{P_{i\tau}} \geq 0$ fir all i, then the original LP is unbounded and the maximum feasible step size is infinitely large. The algorithm stops in that very case.

If a feasible Pivot element is found the following update rules can be used, in order to obtain the new Simplex tableau.

$$\begin{array}{r|c|c} ST_n & x_N & 1 \\ \hline x_B = & P & p \\ \hline z = & q^T & q_0 \end{array} \longrightarrow \begin{array}{r|c|c} ST_{n+1} & x_{\bar{N}} & 1 \\ \hline x_{\bar{B}} = & \bar{P} & \bar{p} \\ \hline z = & \bar{q}^T & \bar{q}_0 \end{array}$$

$$
\begin{aligned}
I_{\bar{N}} &:= N\backslash\{\tau\}\cup\{\sigma\} \\
I_{\bar{B}} &:= B\backslash\{\sigma\}\cup\{\tau\}
\end{aligned}
$$

$$
\begin{aligned}
\bar{P}_{\sigma\tau} &:= \frac{1}{P_{\sigma\tau}} \\
\bar{P}_{\sigma j} &:= -\frac{P_{\sigma j}}{P_{\sigma\tau}} && j\in N\ \backslash\{\tau\} \\
\bar{P}_{i\tau} &:= \frac{P_{i\tau}}{P_{\sigma\tau}} && i\in B\backslash\{\sigma\} \\
\bar{p}_{\sigma} &:= -\frac{p_{\sigma}}{P_{\sigma\tau}} \\
\bar{p}_{i} &:= p_i-\frac{p_{\sigma}}{P_{\sigma\tau}}\cdot P_{i\tau} && i\in B\backslash\{\sigma\} \\
\bar{q}_{\tau} &:= \frac{q_{\tau}}{P_{\sigma\tau}} \\
\bar{P}_{ij} &:= P_{ij}-\frac{P_{\sigma j}}{P_{\sigma\tau}}\cdot q_{\tau} && i\in B\backslash\{\sigma\}, j\in N\backslash\{\tau\} \\
q_{j} &:= q_j-\frac{P_{\sigma j}}{P_{\sigma\tau}}\cdot q_{\tau} && j\in N\backslash\{\tau\} \\
\bar{q}_{0} &:= q_0-\frac{p_{\sigma}}{P_{\sigma\tau}}\cdot q_{\tau}
\end{aligned}
$$

These are in fact just equivalent alterations of the original equation system, hereby swapping a basic and a non-basic variable. Thus, for given LP and its equation system we can uniquely determine its state by just its basic variables. The precise calculations and visualizations are given in the appendix.

Lemma 2. *A Simplex tableau and thus its solution is optimal iff $p \geq 0$ and $q \geq 0$, thus being primal and dual feasible.*

The algorithm eventually stops with respect to the above lemma.

The way the Simplex algorithm works can be observed in a visualization of its feasible space1. The first feasible basic solution is marked in black, the second in blue and the third in red. At first we are able to discover that the formerly mentioned quite abstract definition of a vertex corresponds to what we would also call a vertex in real life. Additionally, we can observe that the algorithm is moving along the edges of the polytope hopping from vertex to vertex along the feasible direction, defined by the objective function.

Eventually, what happens if don't chose row and column as given by Simplex update rules? By swapping a basic and a non-basic index, we obtain a new set of basic variables. if this state is not feasible, we are off the polytope, and not walking on the edges of it anymore. If we reach a feasible vertex, then we just took step along an edge in a direction possibly different from the direction determined by the objective function. We could do so by running a Simplex step and then swapping the two indices back. The latter step has to be feasible, as it was in the opposite direction. We are thus now free to roam along the polytope.

5.2.3 Complexity, Degeneracy and other Properties

Eventually, we want to briefly discuss the power and the significance of this very algorithm, especially with respect to our original topic, the VEP.

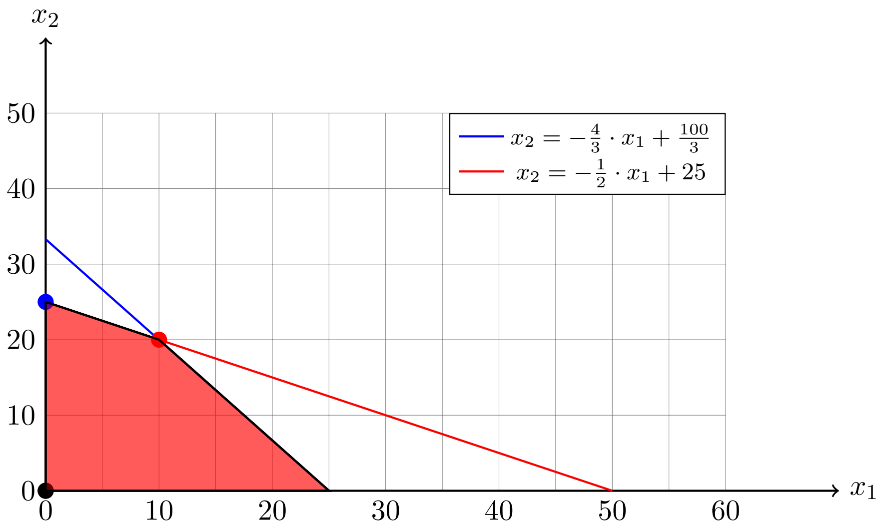

Figure 1: Visualization of the simplified example problem

For bounded, non-*degenerate* polytopes, the original Simplex algorithm terminates and returns the optimal solution in polynomial time (on average). However, a polyhedron is not always non-degenerate as we will see with the following definition.

Definition 11. *(Degeneracy)*
If a vertex $v \in G$ solves more than m inequalities as equations, then we will call this vertex degenerate*.*
If a polyhedron has at least one degenerate vertex, then we will call this polyhedron degenerate*.*

the Simplex method struggles when multiple vertices have the same objective value, while also being mutually adjacent. This is the case for degenerate vertices, as we have to interpret one of the basic variables as 0, too. Thus, multiple states of our Simplex tableau refer to the same vertex, which could lead the algorithm to circle along the basic solutions. However, we can simply introduce an order among the vertices, stating in which sequence they will be chosen, e.g. a lexicographic order of their corresponding basic indices set. Thus, the Simplex algorithm becomes deterministic and terminates in every case.

The implications of Bland's rule are discussed in Theorem 4.

Theorem 4. *(Bland's rule)*
If we use Bland's rule for choosing the Pivot element for the Simplex step, then these updated Simplex steps will always yield a primal feasible basic solution. If Bland's rule is not applicable, then the basic solution is dual feasible and thus optimal.

Additionally, there are much simpler ways to choose a Pivot element. The Criss-Cross rule is choosing its row in the Simplex tableau, not by minimizing over the quotients as in 10, but by choosing purely combinatorial. Thus, the updated Simplex tableau doesn't have to one maximizing the difference of objective function, as e.g., given in Bland's rule.

Lemma 3. *If (P) is solvable, then there is a vertex of G which solves (P).*

Thus, with both rules we will deterministically find the optimal basic solution and its vertex.

Proposition 1. *Let G be a polytope. If we perform the Simplex algorithm with the update strategy*

- *Bland's rule, then each generated tableau will be primal feasible and the procedure leads to an optimal basic solution.*

- *Criss-Cross rule, then the procedure leads to an optimal basic solution.*

Both rules add determinism and thus yield a unique successor for each basic solution v. Concatenating the successor edges, we gain a unique path from each basic solution to the optimal one. As this graph cannot have any cycles, these paths build a spanning tree of all vertices of G, rooted in the optimal vertex.

The degeneracy of the optimal vertex however, cannot be solved by neither Bland's or Criss-Cross rule. The graph thus becomes a spanning forest, as the Simplex algorithm stops when reaching an optimal basic solution. Fortunately, we are able to find all optimal basic solutions by a simple search strategy explained later on. If we are given all optimal vertices, we can construct an artificial new vertex, from which all optimal basic solutions are reachable. We do so by

6 Avis and Fukudas Algorithm

While these properties of LPs are fairly interesting, how does this help to eventually solve the VEP? Given a polytope G, we introduce a random but fixed linear objective function $f(x) := c^T \cdot x$ with $c^T \neq 0$, that defines any gradient along G. This leads to the previously mentioned spanning tree of G.

It was the idea of Avis and Fukuda, published in the discussed paper, to start a reverse search over this spanning tree to find all vertices, starting from the optimal node. But, the Simplex algorithm only teaches us how to move along the gradient of f, and thus only gives as a unidirectional, directed path from any vertex v to the root vertex, but not vice versa.

Definition 12. *Let v be a basic feasible solution (vertex).*
Let (τ, σ) be the pivot obtained by applying any *rule to v yielding v'.*
We call (σ, τ) the reverse pivot for v'.

By simply pivoting with the same two indices, we can revert a previously done Bland's rule or Criss-Cross rule step.

This helps us to solve the main issue and major contribution of this paper:

Given a vertex v, how do we find all vertices v' that have v as its successor?

As we can only determine successors and not predecessors of vertices, this problem still remains. Avis and Fukuda were able to resolve this issue by a simple brute force approach: We therefor check all neighbours v' of a given vertex v, and check whether Bland's rule takes us back to the original basic solution. In order to do so, we first have to propose the characterization of a neighbour.

Definition 13. *(Neighbour)*
For a given vertex v defined by its set of basic variables B, we can define adjacency *as*

$$Adj(B,i,j) := \begin{cases} \emptyset & P_{ij} = 0 \\ \emptyset & (B \cup \{i\}) - \{j\} \text{ } \textit{infeasible tableau} \\ (B \cup \{i\}) - \{j\} & \textit{feasible tableau} \end{cases}$$

The whole neighbourhood of a vertex is given by

$$\textsc{neighbours}(B,N,P) := \{Adj(B,i.j) | \textit{for } i \in N, j \in B \textit{ if } Adj(B,i,j) \neq \emptyset\} \tag{11}$$

As distinguishing feasible and infeasible tableaus or states is simple, we can easily assemble all feasible neighbours of v. We now formulate the above informal description of the procedure into the following algorithm.

Algorithm 2 Reverse-search the spanning tree

```
SEARCH(B,N,P):
for all i, j in NEIGHBOURS(B, N, P) do
    if REVERSE(B, N, P, i, j) then
        PIVOT(B, N, P, i, j)
        if LEX_MIN(B, N, P) then
            print(B)
        end if
        SEARCH(B, N, P)
        SELECT-PIVOT(P)
        PIVOT(B, N, P, i, j)
    end if
end for
```

The function REVERSE(B, N, P, i, j) checks whether (i, j) is a reverse pivot of the state given by B. Additionally, we introduce the method LEX_MIN(B, N, P), which is helpful for the reduction of degenerate vertices. This method checks the current basic solution if it is the minimum with respect to a predefined, lexicographic ordering, among all basic solutions of the current vertex. As the algorithm will visit every basic solution, we thus ensure that the vertex gets printed only once. SELECT-PIVOT hereby corresponds to any of the previously mentioned pivoting rules.

Starting at the root node, the algorithm checks all any of the neighbour vertices for existence of reverse pivots. If that is the case, the current node gets printed. We then start a recursive search for more vertices from the previously discovered node. While in this algorithm formulation, find it to be a depth first search, any search technique is viable. If the algorithm has finished the SEARCH sub-procedure, it jumps back to its successor with a simple pivot step. Starting from any first feasible solution, found by usage of the first phase of the Simplex method, we can always find the initial, optimal, root vertex applying the Bland's rule iteratively.

Eventually, we reached our goal of finding all vertices for our example problem, and are thus free to cook delicious dishes.

6.1 Complexity and other properties

The algorithm gains the following properties derived from the previously described sub-procedures. At first the method does not need any additional storage beyond the input data, as all calculations can be run on the input Simplex tableau. Second, by construction the output will be free of duplicates. Every basic solution will be visited multiple times, but due to the spanning tree and handling of degenerate cases, only output once. Third, the algorithm is extremely simple, requires no complicated data structures, as everything can be computed on the tableau itself. Fourth, the procedure immanently handles all degenerate cases for both optimal and regular tableaus. Regarding time complexity, reverse search is dependent on the amount of vertices, and thus output sensitive for non-degenerate cases. Eventually, given by its recursive search nature and the independence of the subtree searches, the method can be parallelized efficiently and easily.

Discussing the complexity in more detail, we assume a polyhedron $G \subset \mathbb{R}^d$ with v vertices, bounded by n inequalities. As the dimensionality of G is given by d, this is also the amount of necessary non-basic variables. Hence, the Simplex tableau consists of $(n+1) \times (d+1)$ entries

added to the constant amount of pointers necessray for the recursion, which reduces to $\mathcal{O}(nd)$ in space complexity.
Observing time complexity, the algorithm has to visit and check all v vertices for the REVERSE property. As there are $|B| \cdot |N|$ indices pairs for possible neighbours, we have to repeat the search and revert check vdn times. Hence, the time complexity reduces to $\mathcal{O}(vdn)$ for non-degenerate polyhedra. When including degenerate cases the complexity grows to $\mathcal{O}\left(dn(n+d)\binom{n-2}{d}\right)$. However, for unbounded polyhedra the complexity becomes NP-hard. These bounds also hold for the FEP, given its duality to the VEP.

Up until now we treated polyhedra and arrangements similarly, but with respect to time complexity their performance diverges. For a simple, i.e. a non-degenerate, arrangement with n hyperplanes in $\mathbb{R}^d$ and v vertices, we have a complexity of $\mathcal{O}(n^2dv)$. This result corresponds to the degenerate complexity for polyhedra with v substituting $\binom{n}{d}$, and thus the number of distinct/non-degenerate states checked by LEX_MIN. We are not able to reduce the complexity further due to the lacking convexity and the multiple possible multiple cells of the given arrangement.

7 Future Work based on this

this paper presented not only an approach to three important problems in computational geometry, but also sparked research in many different directions. First, its so fundamental yet so elegant, taht is cited by nearly every book on polytopes and computational geometry. Second, the authors themselves built various publications on top, as they investigated the reverse search technique (Avis, Fukuda 1996)[4] and double description methods (Fukuda, 1995)[5]. These paper in return were cited 500 times each.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest und Clifford Stein. *Introduction to Algorithms.* Third Edition. The MIT Press, 2009.

[2] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all verticesof a polyhedron is hard.Discrete and Computational Geometry, 39(1):174–190, 2008.

[3] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, The Double Description Method, Annals of Mathematical Studies, vol. 8, Princeton University Press, Princeton, N J, 1953.

[4] Avis, D., & Fukuda, K. (1996). Reverse search for enumeration. Discrete Applied Mathematics, 65(1–3), 21–46. https://doi.org/10.1016/0166-218x(95)00026-n

[5] Fukuda and Prodon, Double Description Method revisited, Franco-Japanese and Franco-Chinese Conference on Combinatorics and Computer Science, 1995

8 Appendix