Part 17

Linear programming 2: A naïve solution algorithm

$$\begin{aligned} \text{minimize} \quad & c^T x \\ & Ax \geq b \end{aligned}$$

A naïve algorithm

Theorem: A polyhedron can only have finitely many vertices.

Corollary: One (simplistic) way to find a solution to a linear program is the following procedure:

1. Convince ourselves that the linear program has a bounded solution
2. Find all *basic solutions*
3. Among these, identify all *feasible* basic solutions by testing which of the basic solutions satisfy all constraints. These are the vertices of the feasible set
4. Among these, find the vertex (feasible basic solution) or vertices that have the lowest value of the objective function. These are the solution(s) of the problem

A naïve algorithm

Practical implementation of step 2:
A basic solution of a problem with constraints

$$Ax \geq b, \quad \text{or equivalently} \quad a_i^T x \geq b_i, i = 1 \ldots m$$

is a point x at which n linearly independent constraints are active. (In addition to possibly more constraints that then need to be linearly dependent on the previous ones.)

One way to enumerate all basic solutions is by enumerating all subsets of n constraints among the total of m constraints:

- Take all possible selections I of n indices within the set $[1,m]$
- For each I see if the constraints are linearly independent. If so, find the (unique) point x at which

$$a_i^T x = b_i \quad \forall i \in I$$

This is a basic solution.

A naïve algorithm

Practical implementation of step 2 – example:

If we have 3 variables $x=\{x_1,x_2,x_3\}$ and 8 constraints

$$Ax \geq b, \quad \text{or equivalently} \quad a_i^T x \geq b_i, i=1...8$$

then we need to

- try first the set $I=\{1,2,3\}$
- see if the 3x3 matrix $A_I=\begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix}$ has full rank
- If so, then the equation $A_I x_I = b_I$ is unique and x_I is a basic solution
- Continue with the sets $I=\{1,2,4\}, \{1,2,5\}, ..., \{6,7,8\}$ and do the same steps

A naïve algorithm

Practical implementation of step 3:
Now that we have a basic solution *x*, we need to determine which of those are feasible.

By construction, we already know that

$$a_i^T x = b_i \quad \forall i \in I$$

but we also have to check the remaining *m*-*n* constraints:

- Go through all indices $i \notin I$
- If for any of these indices $a_i^T x < b_i$ then this basic solution is infeasible, i.e. it can not be a feasible basic solution and therefore not be a vertex. We can discard this basic solution
- If the basic solution turns out to be feasible with regards to all other constraints, then it must be a vertex

A naïve algorithm

Practical implementation of step 4:
Now that we have a feasible basic solution x, we need to determine which one is the best with regard to the objective function.

To do this:

- For every set of n indices I compute x_I as the basic solution
- If it turns out to be feasible, compute $f(x_I)=c^T x_I$
- If this value $f(x_I)$ is bigger than the previously smallest one seen, then forget about this feasible basic solution and move on to the next set of n indices
- If this value $f(x_I)$ is smaller than the previously smallest one seen, then save $f(x_I)$ and x_I for later comparison and move on to the next set of n indices

A naïve algorithm

Assessment of the algorithm:

- The algorithm works and finds the solution if there exists a bounded solution
- The algorithm is unaffected by degeneracy
- The algorithm is slow because it needs to test *every* vertex of the feasible region
- Since the number of vertices in general grows combinatorically with the number of variables and constraints, the run time of the algorithm grows exponentially as

$$\binom{m}{n}\left(n^3+(m-n)n\right)\approx(2.5\gamma)^n \qquad \text{if } m=\gamma n$$

- Such algorithms are not suited for practical, large-scale problems with thousands or millions of variables and constraints

Part 18

Linear programming 3: Dantzig's *simplex algorithm*

$$\begin{aligned} \text{minimize} \quad & c^T x \\ Ax &= b \\ x &\geq b \end{aligned}$$

The idea

Instead of enumerating and testing *all* vertices, we should:

- Start with a feasible basic solution (vertex)
- Tests its neighbors and go to one with a lower objective function value
- Since the objective function values are a decreasing sequence, cycling is not possible; since there are only finitely many vertices, the algorithm must terminate in a finite number of steps
- Since we only accept vertices with lower objective function values, we hope that we need to visit far fewer than all vertices

This is the basic idea of Dantzig's *simplex* algorithm

Preliminary considerations 1

Theorem:
Let the feasible set of a linear program in standard form be described by the equations

$$P=\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\}$$

where the matrix A does not have full row rank (i.e. its rows are linearly dependent).
If P is not empty, then there exists a matrix with full row rank so that

$$Q=\{x\in\mathbb{R}^n:\quad \tilde{A}x=b, \tilde{A}\in\mathbb{R}^{m'\times n}, m'<m\leq n,\quad x\geq 0\}$$

and $Q=P$.

Due to this equivalence, we will in the following always assume that A has full row rank.

Preliminary considerations 2

The feasible sets of linear programs in standard form are also polyhedra and are described by the equations

$$P=\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\}$$

Then at any feasible basic solution (vertex of P) the following holds true:

- all m equality constraints are active
- at least n-m variables x_i are zero
- if a basic solution is non-degenerate, exactly n-m variables are zero

Standard form is so convenient because we don't just know that *n-m* inequalities are active, but can associate them with vector components!

Preliminary considerations 3

Definition:

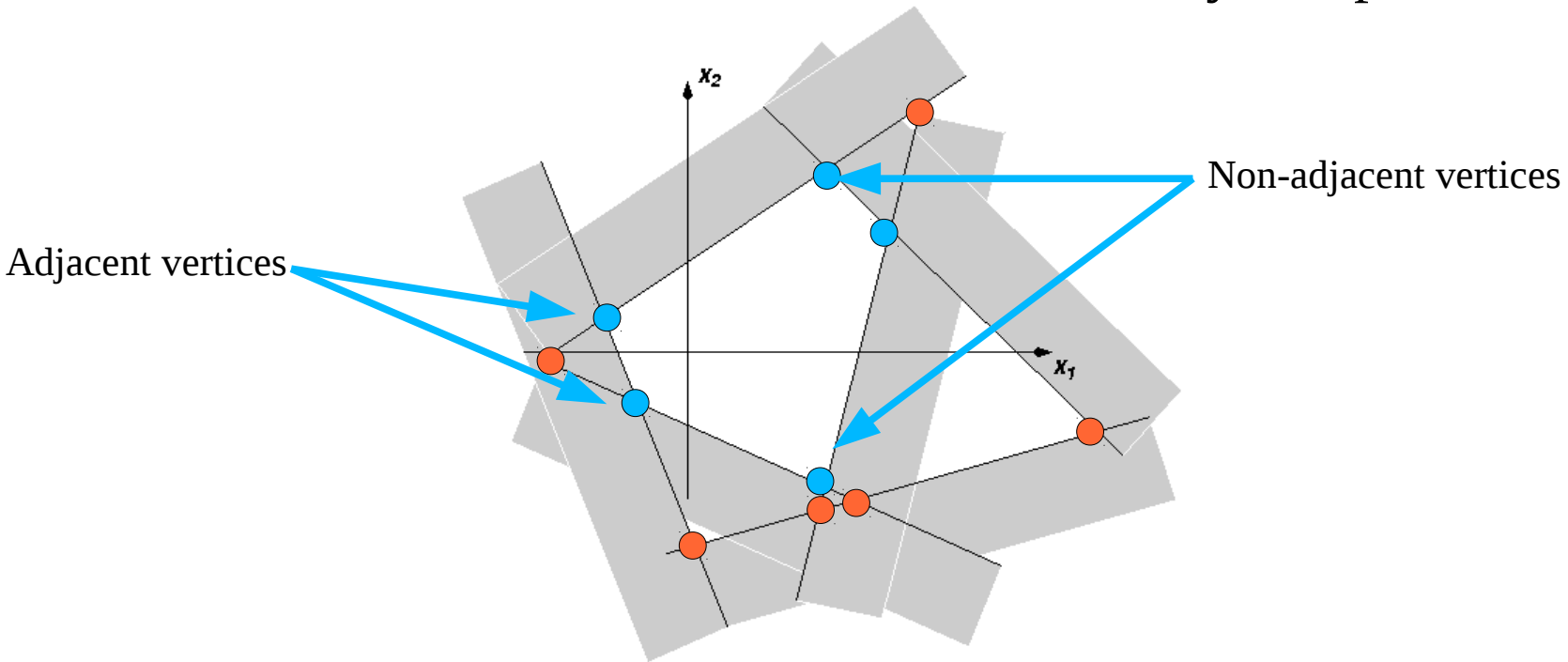
Let $P \subset \mathbb{R}^n$ be a polyhedron. Let $p_1, p_2 \in \mathbb{R}^n$ be two basic solutions of P. We call them *adjacent* if

$$\{a_i : i \in I(p_1)\}$$

and

$$\{a_i : i \in I(p_2)\}$$

contain a common set of *n-1* vectors that are linearly independent.



Preliminary considerations 3

In particular, for standard form:
Since equality constraints always have to be active, every feasible basic solutions of a polyhedron in standard form

$$P=\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\}$$

must have *m* active equality constraints, and

- exactly *n*-*m* variables x_i that are zero (if the basic solution is not degenerate)
- or more than *n*-*m* variables x_i that are zero (if the basic solution is not degenerate).

In the non-degenerate case, two basic solutions are adjacent if they differ in exactly one pair of variables x_i that are zero/nonzero at one vertex and nonzero/zero at the other!

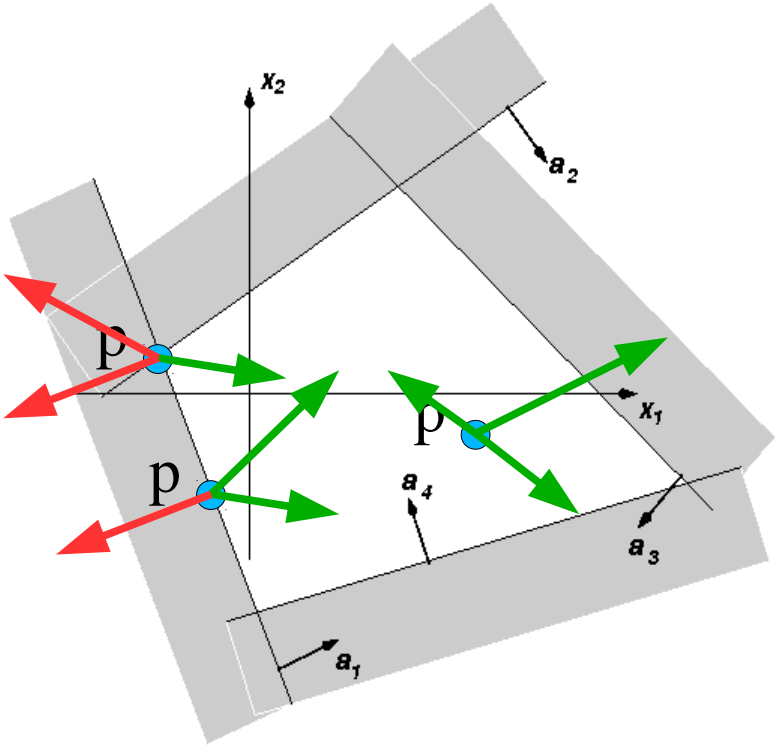
Preliminary considerations 4

Definition:

Let x be a point in a polyhedron P. Then we call a vector d a *feasible direction* if

$$\exists \theta > 0: \; x + \theta d \in P$$

Example:



feasible directions

infeasible directions

Preliminary considerations 4

In particular:

Let *p* be a non-degenerate vertex of a polyhedron *P* described in standard form:

$$P=\left\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\right\}$$

Let *I(p)* be the active set of constraints at *p*. Then any *feasible direction d* needs to satisfy the following conditions:

$$\begin{aligned} Ad&=0\\ d_i&\geq 0\quad \forall\, i+m\in I(p)\end{aligned}$$

Preliminary considerations 4

Conversely:
Let I, $\#I=n$ be a set of indices. Assume the associated constraints are linearly independent. Then I describes a vertex p of a polyhedron

$$P=\left\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\right\}$$

If the vertex *is not* degenerate, then any direction that satisfies

$$\begin{aligned} Ad&=0\\ d_i&\geq 0\quad \forall\, i+m\in I(p)\end{aligned}$$

is feasible. If the vertex *is* degenerate, then we have to require that

$$\begin{aligned} Ad&=0\\ d_i&\geq 0\quad \forall\, i+m\in I(p)\\ d_i&\geq 0\quad \forall\, i+m\notin I(p), x_i=0\end{aligned}$$

Note: These relations can be used to test whether a proposed direction is feasible or not.

The simplex algorithm, non-degenerate case

The simplex algorithm works on standard form:

$$P=\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\}$$

At every step of the simplex algorithm, the current state is described by the following pieces of information:

- A set of indices H, $\#H=m$, called the *basis*. H describes the variables that are *not* bound by the constraints and so is somewhat complementary to the set of active indices I.
- H defines a *basis matrix* $B=A_H$ that consists of the *columns* of A listed in H. B is the "interesting" part of the matrix A_I.
- H defines a basic solution x of a polyhedron (which in the algorithm will always be feasible) that satisfies

$$Bx_H=b$$
$$x_{H^c}=0$$

Due to non-degeneracy, $x_H>0$ for each vector element.

The simplex algorithm, non-degenerate case

Why bases instead of active sets:

Let the polyhedron be described by

$$P=\left\{x\in\mathbb{R}^n:\quad Ax=b, A\in\mathbb{R}^{m\times n}, m\leq n,\quad x\geq 0\right\}$$

Then at every (non-degenerate) basic solution we have an active set I with exactly n elements. These are:

- The indices $1...m$ corresponding to equality constraints
- A subset of size $(n\text{-}m)$ of the indices $m+1...m+n$ corresponding to the positivity constraints

The linear system that describes the basic solution is therefore:

$$\begin{aligned} A\,x&=b\\ x_{I_i}&=0\quad i=m+1...n\end{aligned}$$

x therefore consists of two parts: components x_H that are not necessarily zero, and x_{H^c} that must be zero. Therefore, in the first equation, only columns listed in H participate, i.e. the basis matrix B.

The simplex algorithm, non-degenerate case

The main idea of the simplex algorithm:

Let H,B be the current basis and basis matrix, and x be a non-degenerate feasible basic solution defined by H.

- To move from x_H to another vertex, we need to release one non-basic variable j from its constraint $x_j=0$ and make it positive.

- Our search direction should therefore be

$$d_j=1$$
$$d_i=0 \quad i\notin H, i\neq j$$

- The basic components need to satisfy

$$A(x+d)=b \quad \rightarrow \quad Ad=0 \quad \rightarrow \quad Bd_H+A_j=0 \quad \rightarrow \quad d_H=-B^{-1}A_j$$

where A_j denotes the jth column of A.

- The vector d so defined is called the *jth basic direction*.

The simplex algorithm, non-degenerate case

Theorem:

Let H,B be the current basis and basis matrix, and x be a non-degenerate feasible basic solution defined by H.

Then for every $j \notin H$ the direction defined by

$$d_j = 1$$
$$d_i = 0 \quad i \notin H, i \neq j$$
$$d_H = -B^{-1} A_j$$

is feasible.

Note 1: If x is degenerate, then d is a feasible direction if and only if

$$(d_H)_i \geq 0 \quad \forall i \in H, x_i = 0$$

Note 2: Feasibility should not be a surprise – we still have *(n-1)* constraints that are active. d is constructed to lie in this 1d subspace.

The simplex algorithm, non-degenerate case

Theorem:

Let H,B be the current basis and basis matrix, and x be a feasible basic solution defined by H.

Then for every $j \notin H$ the jth basic direction is a direction of descent of the objective function if the *reduced cost* satisfies

$$\bar{c}_j = c_j - c_H^T B^{-1} A_j < 0$$

The simplex algorithm, non-degenerate case

Theorem:

Let H,B be the current basis and basis matrix, and x be a feasible basic solution defined by H. Then:

- If $\bar{c}_j = c_j - c_H^T B^{-1} A_j \geq 0 \quad \forall\, j \notin H$ then x is optimal

- If x is optimal and non-degenerate, then $\bar{c}_j \geq 0 \quad \forall\, j \notin H$

Note: The first condition can be used to test whether a vertex x is optimal – we only need to compute all reduced costs!

The simplex algorithm, non-degenerate case

Line search:

Let H,B be the current basis and basis matrix, and x be a non-degenerate feasible basic solution defined by H.

Let d be a feasible basic direction. Then:

- $x+\theta d$ satisfies $(n-1)$ constraints for sufficiently small step lengths θ.
- $x+\theta d$ is feasible for

$$\theta \leq \theta^* = \min_{i \in H,\, d_i < 0} \left(-\frac{x_i}{d_i} \right)$$

The simplex algorithm, non-degenerate case

Algorithm:

Let H,B be the current basis and basis matrix, and x be a non-degenerate feasible basic solution defined by H. Then perform the following steps:

Let $j=1...n$, $j \notin H$

Compute $d_H = -B^{-1}A_j, \quad \bar{c}_j = c_j - c_H^T B^{-1} A_j$

If $\bar{c}_j < 0$ then

- compute $\theta^* = \min_{i \in H, d_i < 0} \left(-\frac{x_i}{d_i} \right)$ and let l be the index for which the minimum is attained
- set

$$x_j \leftarrow \theta^*, \quad x_H \leftarrow x_H + \theta^* d_H$$
$$H \leftarrow H \setminus \{l\} \cup \{j\}$$
$$B_l \leftarrow A_j$$

- start over

Try the next j. If no j allows has negative reduced costs, then we have a solution.

The simplex algorithm, non-degenerate case

Note:

If all components of *d* turn out to be positive, then we have a direction in which every point is feasible and we can choose $\theta^* = \infty$. This means that the problem has no bounded solution at the point where we compute the step length we have already determined that *d* is a direction of descent.

Theorem:

Assume the basic matrix $B = \left(A_{H_1} A_{H_2} \ldots A_{H_m} \right)$ at the beginning of the iteration has full rank. Then the new basic matrix

$$B = \left(A_{H_1} A_{H_2} \ldots A_j \ldots A_{H_m} \right)$$

also has full rank.

The simplex algorithm, non-degenerate case

Theorem:

The algorithm just outlined terminates after finitely many steps with one of the following results:

- If all reduced costs $\bar{c}_j$ are non-negative, then the current vertex is a solution of the minimization problem
- If at a vertex at least one of the reduced costs $\bar{c}_j$ is negative but the corresponding search direction satisfies $d>0$, then the linear problem is unbounded from below and has no bounded solution.

The simplex algorithm, non-degenerate case

Note: In our algorithm, we test

- Is one of the reduced costs $\bar{c}_j$ negative
- If so, let *j* enter the basis (i.e. release its constraint and make it a free variable)

Question:

What do we do if the reduced costs are negative for more than one index?

We could choose any variable with a negative reduced cost, but maybe some strategies will lead to algorithms that require fewer iterations than others.

The simplex algorithm, non-degenerate case

Question: What do we do if the reduced costs are negative for more than one index?

Answer: There are many *pivoting* strategies, for example we could

- Choose that index j for which the reduced cost is the most negative
- Choose that index j for which $\theta^* \bar{c}_j$ is the most negative
- Try to choose an index j that has not recently been chosen
- ....
- Bland's rule: Take the first index j for which the reduced cost is negative

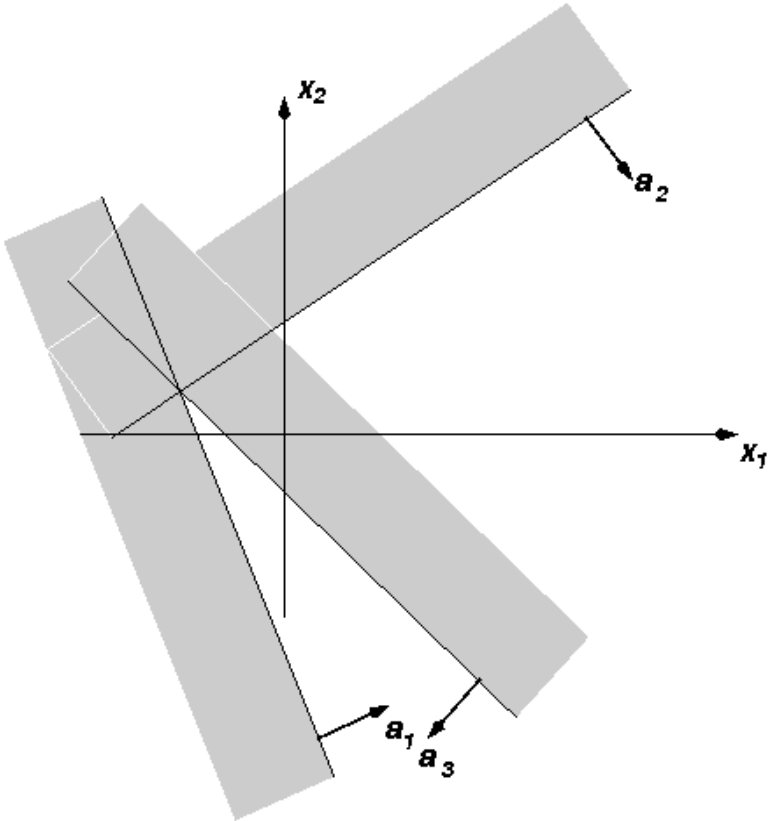
Note: More complex strategies often reduce the number of iterations at the cost of more expensive iterations. Bland's rule is a good choice to avoid *cycling* in the degenerate case.

The degenerate case

Two things can happen in the degenerate case:

- We want to release constraint j and let x_j enter the basis but we can't go into direction $d=-B^{-1}A_j$ because we immediately hit a previously active constraint that was not part of the basis H.

  In other words, we find that $\theta^*=0$. What do we do?



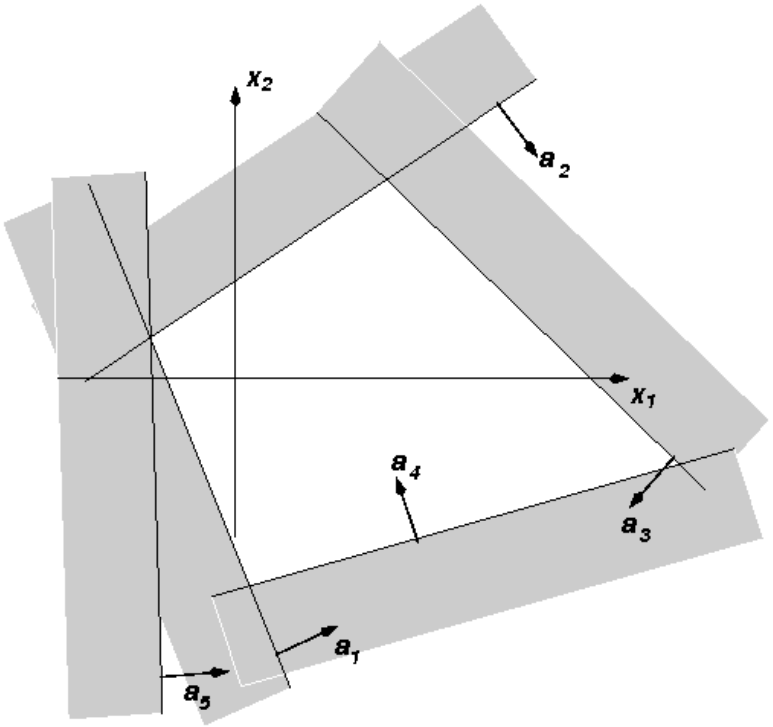
Example (not for standard form linear problems): Constraints 1 and 2 are active. We want to release constraint 1 but we can't move away from it.

The degenerate case

Two things can happen in the degenerate case:

- We let x_j enter the basis which then has *(n-1)* active constraints. We move in direction $d=-B^{-1}A_j$ but at θ^* we find more than one new constraint.

  In this case, which of these constraints should exit the basis?



Example (not for standard form): Constraints 2 and 3 are active. We want to release constraint 3 and move along constraint 2 but then both constraints 1 and 5 become active.

The degenerate case

Case 1: We find that $\theta^* = 0.$

In this case, we know that more than *n* constraints are active at the current vertex, i.e. some of the basic ("free") variables in *H* are zero.

The question is then which of these variables to throw out of the basis in response to letting x_j enter the basis without taking a step that decreases the objective function?

The question is important because we want to avoid *cycling*, i.e. returning to the same basis after a number of steps without reducing the objective function.

Answer: There are a number of *pivoting* strategies. The simplest is Bland's rule – if there are multiple constraints that become active, take the one with the smallest index.

The degenerate case

Case 2: We find that $\theta^* > 0$ but that more than one constraint becomes active.

The question is then which of the variables whose constraints become active to throw out of the basis (i.e. let them “exit the basis”) in response to letting x_j enter the basis?

Answer: There are a number of *pivoting* strategies. The simplest is Bland's rule – if there are multiple constraints that become active, take the one with the smallest index.

The degenerate case

Theorem:

Using Bland's rule for selecting

- which variable will enter the basis
- which variable will exit the basis if there are multiple that are eligible

avoids the problem of cycling and therefore guarantees that the algorithm terminates in finite time.

Starting the simplex method

Problem: The simplex algorithm needs to start from a feasible basic solution.

Unfortunately, finding any vertex is almost as expensive as finding the best one.

Typical strategy: The “two-phase simplex method”

Starting the simplex method

Starting point: Consider the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

Without loss of generality, we can assume that $b \geq 0$. We seek a feasible vertex of the feasible set and a corresponding basis.

Consider now the auxiliary problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^m} \quad & (1,1,\ldots,1)^T y \\ & Ax + y = b \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

Notes:

- For no (feasible) choice of *y* can the objective function of the auxiliary problem be negative.
- It is zero if *x* is a feasible point of the original problem.
- It is positive if there is no feasible point of the original problem.

Starting the simplex method

Consider the auxiliary problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^m} \quad & (1,1,\ldots,1)^T y \\ & Ax + y = b \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

We can solve this problem using the simplex algorithm as discussed, starting with the feasible vertex *(x=0, y=b)*. The algorithm terminates with either of these outcomes:

- *The objective function is positive:* The original problem has no feasible point
- *The objective function is zero:* Then, *y=0* and *x* is a feasible vertex with respect to the original problem.
  Furthermore, at least *n* variables among the *x*,*y* are at their bounds, and at most *m* variables are greater than zero

Starting the simplex method

However, we need more to start the simplex algorithm on the original problem:

We need a basis H, which then implies the location of the vertex as well as the basis matrix B.

Can we use the final basis H_{aux} (consisting of m free variables) of the simplex algorithm applied to the auxiliary problem?

- If the vertex we find is non-degenerate, then all m entries in H_{aux} are components of x (because $y=0$) and we can use $H=H_{aux}$
- If the vertex is degenerate, then H_{aux} contains variables that are zero at the solution and could contain auxiliary variables y.
  We then need a procedure to “drive artificial variables out of the basis”.

Starting the simplex method

Driving artificial variables out of the basis:

H_{aux} has m entries but some of them correspond to auxiliary variables and only $k<m$ non-artificial entries.

We need a basis H with m non-artificial entries. We can let currently non-basic variables x_i (for which $x_i=0$) enter this basis, but we need to make sure that the corresponding basis B retains full rank. The following procedure guarantees this:

While $k<m$:

- Let $l>k, l\leq m$ be an index in H_{aux} so that $(H_{aux})_l$ corresponds to an artificial variable
- Choose $j<m$ so that $\left(B_{aux}^{-1} A_j\right)_l \neq 0$
- Replace $H_{aux} \leftarrow H_{aux} \setminus \left\{\left(H_{aux}\right)_l\right\} \cup \{j\}$ and re-assemble B_{aux}.

The two-phase simplex algorithm

1. Remove linearly dependent constraints from the matrix A
2. Multiply constraints as necessary so that $b \geq 0$
3. Introduce artificial variables $y \in \mathbb{R}^m$ and solve the auxiliary problem
4. If the objective function at the solution is positive the original problem does not have a feasible solution. Terminate.
5. Given H_{aux}, B_{aux}, drive artificial variables out of the basis until they only contain non-artificial variables
6. Set $H = H_{aux}$, $B = B_{aux}$

7. Solve the original problem using the simplex algorithm

Implementing the simplex method

Naïve implementation:

In a naïve implementation, in each iteration we have to

- Compute B^{-1} $O(m^3)$
- Compute reduced costs $O(m(n-m))$
- Compute a search direction $O(m^3)$
- Compute a step length $O(m)$

Thus, the total effort is $O(m^3+mn)$ per visited vertex.

Better implementations: We can achieve the same result using only $O(m^2+mn)$ operations per visited vertex. This uses, for example, updating rules to compute B^{-1} from the basis matrix used in the previous iteration.

Complexity of the simplex method

Overall complexity: In the best case, each iteration costs $O(m^2+mn)$. How many iterations does one need?

In practice: In most applications, the number of iterations appears to be a small multiple of the number of constraints m.

In theory: For most pivoting rules, examples are known where every single vertex is visited. These examples typically involve the 2^n vertices of the unit cube with n variables and $m=2n$ constraints.

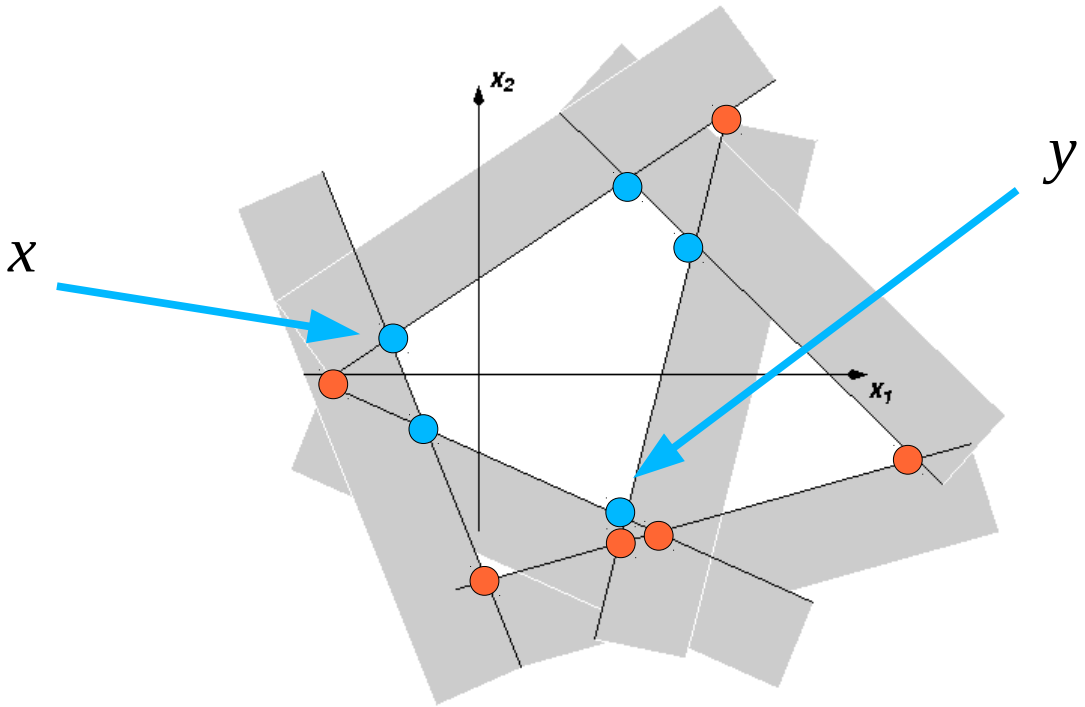
Questions:

- How often does this happen?
- Is this a property of individual algorithms/pivoting rules?
- Is this a property of linear problems?

Complexity of the simplex method

Definition: The distance *d(x,y)* between two vertices *x*,*y* of a polyhedron *P* is the length of the shortest sequence of steps through intermediate vertices.

Definition: The diameter *diam(P)* of a polyhedron *P* is the maximal distance between any two vertices of *P*.



d(x,y)=2

diam(P)=2

Complexity of the simplex method

Corollary: For any pivoting rule, for a bad choice of initial vertex, we always need to expect that we need at least *diam(P)* iterations.

Question: Do we know anything about *diam(P)* for given *n*,*m*?

Complexity of the simplex method

Definition:

Let $$\Delta(n,m) = \max_{\substack{A \in \mathbb{R}^{m\times n} \\ P=\{x\in\mathbb{R}^n:\, Ax\geq b\} \\ P \text{ is bounded}}} \operatorname{diam}(P)$$

$$\Delta_u(n,m) = \max_{\substack{A \in \mathbb{R}^{m\times n} \\ P=\{x\in\mathbb{R}^n:\, Ax\geq b\}}} \operatorname{diam}(P)$$

Corollary:

$$\Delta(n,m) \leq \Delta_u(n,m)$$

$$\Delta(2,m) = \lfloor \frac{m}{2} \rfloor$$

$$\Delta_u(2,m) = m-2$$

Complexity of the simplex method

Hirsch conjecture:

$$\Delta(n,m) \leq m-n$$

Consequence: This would imply that we could hope to find a pivoting rule that always terminates in *O(m)* iterations for bounded problems.

Theorem:

$$\Delta(n,m) \leq \Delta_u(n,m)$$

$$m-(n-\lfloor\frac{n}{5}\rfloor) \leq \Delta_u(n,m) \leq (2n)^{\log_2 m}$$

In other words, the best known upper bound for the diameter of unbounded polyhedra is not exponential, but worse than polynomial in *n,m*.

Complexity of the simplex method

Theorem (Borgwardt 1982):

Consider solving the following problem with the "shadow vertex" variant of the simplex algorithm

$$\min_{x \in \mathbb{R}^n} c^T x$$
$$a_i x \geq 1, \quad i = 1 \ldots m$$

where the vectors c, a_i are chosen randomly in $\mathbb{R}^n \setminus \{0\}$.

Then the *average* number of iterations necessary is less than

$$17 n^3 m^{\frac{1}{n-1}}$$

Note: This does not match practical experience.

Complexity of the simplex method

Theorem (Haimovich, Adler 1982):

Consider solving the following problem with the "shadow vertex" variant of the simplex algorithm

$$\min_{x \in \mathbb{R}^n} c^T x$$
$$a_i x \geq b_i, \quad i = 1 \ldots m$$

where the vectors c, a_i,b are chosen randomly with some assumptions.

Then the *average* number of iterations necessary is less than

$$n \frac{m-n+2}{m+1}$$

Under less stringent assumptions, the number of iterations is bounded by

$$C \min\left\{(m-n)^{2,} n^2\right\}$$