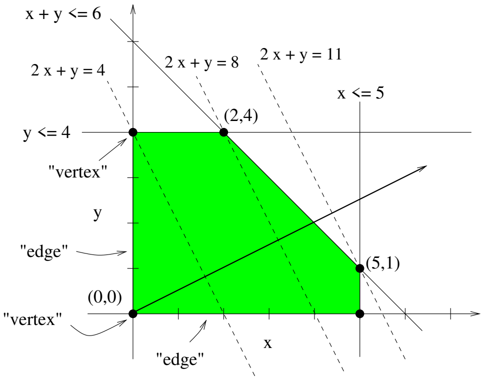
Algorithmic Game Theory and Applications

Lecture 6: The Simplex Algorithm

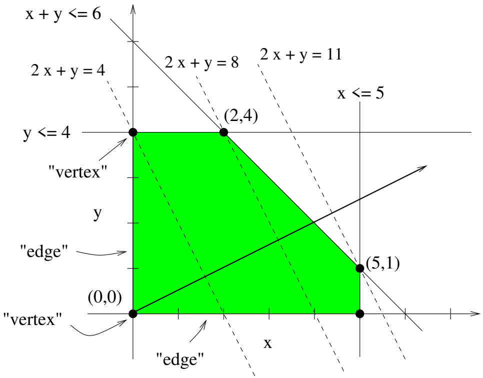
Kousha Etessami

Recall our example



Note: starting at $(0, 0)$, we can find the optimal vertex $(5, 1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

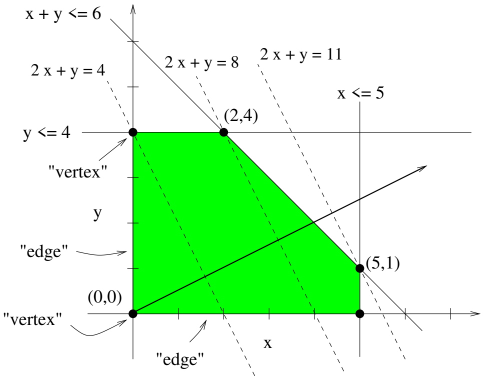
Recall our example



Note: starting at $(0,0)$, we can find the optimal vertex $(5,1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

We don't seem to get "stuck" in any "locally optimal" vertex.

Recall our example



Note: starting at $(0, 0)$, we can find the optimal vertex $(5, 1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

We don't seem to get "stuck" in any "locally optimal" vertex. That's the geometric idea of simplex!

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

geometric idea of simplex

Input: Given $(f, \mathtt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

Key reason: The region $K(C)$ is convex. (Recall: K is convex iff for all $x, y \in K$, $\lambda x + (1 - \lambda)y \in K$, for all $\lambda \in [0, 1]$.)

Fact: On a convex region, a "local optimum" of a linear objective is always a "global optimum".

geometric idea of simplex

Input: Given $(f, \mathtt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

Key reason: The region $K(C)$ is convex. (Recall: K is convex iff for all $x, y \in K$, $\lambda x + (1 - \lambda)y \in K$, for all $\lambda \in [0, 1]$.)

Fact: On a convex region, a "local optimum" of a linear objective is always a "global optimum".

Ok. The geometry sounds nice and simple. But realizing it algebraically is not a trivial matter!

LP's in "Primal Form"

Using the simplification rules from the last lecture, we can convert any LP into the following form:

$$
\begin{array}{ll}
\textbf{Maximize } c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d & \\
\textbf{Subject to:} & \\
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n & \leq b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n & \leq b_2 \\
\ldots & \ldots \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n & \leq b_m \\
x_1, \ldots, x_n \geq 0 &
\end{array}
$$

Side comment: Carrying along the constant d in the objective $f(x)$ seems silly: it doesn't affect the optimality of a solution. (It only shifts the value of a solution by d.) We keep the constant for convenience, to become apparent later.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\ x_1 + c_2\ x_2 + \ldots + c_n\ x_n + d$

Subject to:

$$\begin{array}{ll}
a_{1,1}\ x_1 + a_{1,2}\ x_2 + \ldots + a_{1,n}\ x_n + y_1 & = b_1 \\
a_{2,1}\ x_1 + a_{2,2}\ x_2 + \ldots + a_{2,n}\ x_n + y_2 & = b_2 \\
\ldots & \ldots \\
a_{m,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{m,n}\ x_n + y_m & = b_m
\end{array}$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\ x_1 + c_2\ x_2 + \ldots + c_n\ x_n + d$

Subject to:

$$
\begin{array}{ll}
a_{1,1}\ x_1 + a_{1,2}\ x_2 + \ldots + a_{1,n}\ x_n + y_1 & = b_1 \\
a_{2,1}\ x_1 + a_{2,2}\ x_2 + \ldots + a_{2,n}\ x_n + y_2 & = b_2 \\
\ldots & \ldots \\
a_{m,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{m,n}\ x_n + y_m & = b_m
\end{array}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\ x_1 + c_2\ x_2 + \ldots + c_n\ x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1}\ x_1 + a_{1,2}\ x_2 + \ldots + a_{1,n}\ x_n + y_1 &= b_1 \\
a_{2,1}\ x_1 + a_{2,2}\ x_2 + \ldots + a_{2,n}\ x_n + y_2 &= b_2 \\
\ldots \qquad\qquad\qquad\qquad &\ \ldots \\
a_{m,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{m,n}\ x_n + y_m &= b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.
2. Picking one such variable, y_i, for each equality, we obtain a set of m variables $B = \{y_1, \ldots, y_m\}$ called a **Basis**.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

$$
\begin{array}{ll}
\multicolumn{2}{l}{\textbf{Maximize } c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d} \\
\multicolumn{2}{l}{\textbf{Subject to:}} \\
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n + y_1 & = b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n + y_2 & = b_2 \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n + y_m & = b_m \\
x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0 &
\end{array}
$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.
2. Picking one such variable, y_i, for each equality, we obtain a set of m variables $B = \{y_1, \ldots, y_m\}$ called a **Basis**.
3. Objective $f(x)$ involves only non-Basis variables.

Let's call an LP in this form a "dictionary".

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$
\begin{array}{ll}
x_{n+1} & = b_1 - a_{1,1}\, x_1 - a_{1,2}\, x_2 - \ldots - a_{1,n}\, x_n \\
x_{n+2} & = b_2 - a_{2,1}\, x_1 - a_{2,2}\, x_2 - \ldots - a_{2,n}\, x_n \\
\ldots & \quad \ldots \\
x_{n+m} & = b_m - a_{m,1}\, x_1 - a_{i,2}\, x_2 - \ldots - a_{m,n}\, x_n \\
x_1, \ldots, x_{n+m} \geq 0 &
\end{array}
$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$.

The objective value is then $f(0) = d$.

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$\begin{aligned}
x_{n+1} &= b_1 - a_{1,1}\, x_1 - a_{1,2}\, x_2 - \ldots - a_{1,n}\, x_n \\
x_{n+2} &= b_2 - a_{2,1}\, x_1 - a_{2,2}\, x_2 - \ldots - a_{2,n}\, x_n \\
\ldots & \quad \ldots \\
x_{n+m} &= b_m - a_{m,1}\, x_1 - a_{i,2}\, x_2 - \ldots - a_{m,n}\, x_n \\
x_1, \ldots, x_{n+m} &\geq 0
\end{aligned}$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$.

The objective value is then $f(0) = d$.

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

Question: How do we move from one BFS with basis B to a "neighboring" BFS with basis B'?

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$\begin{array}{ll} x_{n+1} & = b_1 - a_{1,1}\, x_1 - a_{1,2}\, x_2 - \ldots - a_{1,n}\, x_n \\ x_{n+2} & = b_2 - a_{2,1}\, x_1 - a_{2,2}\, x_2 - \ldots - a_{2,n}\, x_n \\ \ldots & \ldots \\ x_{n+m} & = b_m - a_{m,1}\, x_1 - a_{i,2}\, x_2 - \ldots - a_{m,n}\, x_n \\ x_1, \ldots, x_{n+m} \geq 0 & \end{array}$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$. The objective value is then $f(0) = d$.

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

Question: How do we move from one BFS with basis B to a "neighboring" BFS with basis B'? **Answer:** Pivoting!

Pivoting

Suppose our current dictionary basis (the variables on the left) is $B = \{x_{i_1}, \ldots, x_{i_m}\}$, with x_{i_r} the variable on the left of constraint C_r.
The following pivoting procedure moves us from basis B to basis $B' := (B \setminus \{x_{i_r}\}) \cup \{x_j\}$.
Pivoting to add x_j and remove x_{i_r} from basis B:

1. Assuming C_r involves x_j, rewrite C_r as $x_j = \alpha$.
2. Substitute α for x_j in other constraints C_l, obtaining C'_l.
3. The new constraints C', have a new basis: $B' := (B \setminus \{x_{i_r}\}) \cup \{x_j\}$.
4. Also substitute α for x_j in $f(x)$, so that $f(x)$ again only depends on variables not in the new basis B'.

This new basis B' is a "possible neighbor" of B.
However, not every such basis B' is eligible!

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b_i' remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b'_i remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.
2. The new BFS must improve, or at least must not decrease, the value $d' = f(0)$ of the new objective function. (Recall, all non-basic variables are set to 0 in a BFS, thus $f(BFS) = f(0)$.)

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b'_i remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.
2. The new BFS must improve, or at least must not decrease, the value $d' = f(0)$ of the new objective function. (Recall, all non-basic variables are set to 0 in a BFS, thus $f(BFS) = f(0)$.)
3. We should also check for the following situations:
   (a) Suppose all variables in $f(x)$ have negative coefficients. Then any increase from 0 in these variables will decrease the objective. We are thus at an optimal BFS x^*. Output: Opt-BFS: x^* & $f(x^*) = f(0) = d'$.

   (b) Suppose a variable x_j in $f(x)$ has coefficient $c_j > 0$, and the coefficient of x_j in every constraint C_r is ≥ 0. Then we can increase x_j, and objective, to "infinity" without violating constraints. So, Output: "Feasible but Unbounded".

finding and choosing eligible pivots

- In principle, we could exhaustively check the sanity conditions for eligibility of all potential pairs of entering and leaving variables. There are at most $(n * m)$ candidates.
- But, there are much more efficient ways to choose pivots, by inspection of the coefficients in the dictionary.
- We can also efficiently choose pivots according to lots of additional criteria, or pivoting rules, such as, e.g., "most improvement in objective value", etc.
- There are many such "rules", and it isn't clear *a priori* what is "best".

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$
$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$
$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$
$$x_1, \ldots, x_6 \geq 0;$$

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$
Subject to:

$$
\begin{aligned}
x_4 &= 3 - \,x_1 - 3\,x_2 - \,x_3\\
x_5 &= 4 - 2\,x_1 + \,x_2 - 2\,x_3\\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \,x_3
\end{aligned}
$$

$x_1, \ldots, x_6 \geq 0;$ *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$
\begin{aligned}
x_4 &= 3 - \ \ x_1 - 3\,x_2 - \ \ x_3\\
x_5 &= 4 - 2\,x_1 + \ \ x_2 - 2\,x_3\\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \ \ x_3
\end{aligned}
$$

$x_1, \ldots, x_6 \geq 0;$ *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$
Subject to:

$$\begin{aligned}
x_4 &= 3 - \,x_1 - 3\,x_2 - \,x_3 \\
x_5 &= 4 - 2\,x_1 + \,x_2 - 2\,x_3 \\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \,x_3
\end{aligned}$$

$x_1, \ldots, x_6 \geq 0;$ *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$
Subject to:

$$
\begin{aligned}
x_4 &= 3 - \,x_1 - 3\,x_2 - \,x_3\\
x_5 &= 4 - 2\,x_1 + \,x_2 - 2\,x_3\\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \,x_3
\end{aligned}
$$

$x_1, \ldots, x_6 \geq 0;$ *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$
Subject to:

$$
\begin{aligned}
x_4 &= 3 - \,x_1 - 3\,x_2 - \,x_3 \\
x_5 &= 4 - 2\,x_1 + \,x_2 - 2\,x_3 \\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \,x_3
\end{aligned}
$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.
- The latter constraint is "tightest". So, once x_2 is chosen to enter the basis, the *unique* variable that can leave the basis while maintaining feasiblility is x_6.

example of a simplex pivoting step

$$\begin{array}{ll} \textbf{Maximize} & 2x_1 + 3\,x_2 + 4x_3 + 8 \\ \textbf{Subject to:} & \\ x_4 & = 3 - \,x_1 - 3\,x_2 - \,x_3 \\ x_5 & = 4 - 2\,x_1 + \,x_2 - 2\,x_3 \\ x_6 & = 2 - 2\,x_1 - 4\,x_2 + \,x_3 \end{array}$$

$x_1, \ldots, x_6 \geq 0;$ *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.
- The latter constraint is "tightest". So, once x_2 is chosen to enter the basis, the *unique* variable that can leave the basis while maintaining feasiblility is x_6.
- So we rewrite constraint $x_6 = 2 - 2x_1 - 4x_2 + x_3$ as: (**) $x_2 = \frac{1}{2} - \frac{1}{2}\,x_1 + \frac{1}{4}x_3 - \frac{1}{4}x_6$, and rewrite the objective and other constraints, by replacing x_2 with RHS of (**).

The Simplex Algorithm

Dantzig's Simplex algorithm can be described as follows:
Input: a feasible dictionary;

Repeat

1. *Check if we are at an optimal solution, and if so, Halt and output the solution.*
2. *Check if we have an "infinity" neighbor, and if so Halt and output "Unbounded".*
3. *Otherwise, choose an eligible pivot pair of variables, and Pivot!*

Fact If this halts the output is correct: an output solution is an optimal solution of the LP.
Oops! We could cycle back to the same basis for ever, never strictly improving by pivoting.
There are several ways to address this problem......

how to prevent cycling

Several Solutions:

1. Carefully choose rules for variable pairs to pivot at, in a way that forces cycling to never happen.
   Fact: This can be done.
   (For example, use "*Bland's rule*": For all eligible pivot pairs (x_i, x_j), where x_i is being added the basis and x_j is being removed from it, choose the pair such that, first, i is as small as possible, and second, j is as small as possible.)
2. Choose randomly among eligible pivots.
   With probability 1, you'll eventually get out and to an optimal BFS.
3. "Perturb" the constraints slighty to make the LP "non-degenerate". (More rigorously, implement this using, e.g., the "lexicographic method".)

the geometry revisited

- Moving to a "neighboring" basis by pivoting roughly corresponds to moving to a neighboring "vertex". However, not literally true because several Bases can correspond to same BFS, and thus to same "vertex".

  We may not have any neighboring bases that strictly improve the objective, and yet still not be optimal, because all neighoring bases B' describe the same BFS "from a different point of view".
- pivoting rules can be designed so we never return to the same "point of view" twice.
- choosing pivots randomly guarantees that we eventually get out.
- properly "perturbing" the constraints makes sure every BFS corresponds to a unique basis (i.e., we are non-degenerate), and thus bases and "vertices" are in 1-1 correspondence.

Hold on! What about finding an initial BFS?

- So far, we have cheated: we have assumed we start with an initial "feasible dictionary", and thus have an initial BFS.
- Recall, the LP may not even be feasible!
- Luckily, it turns out, it is as easy (using Simplex) to find whether a feasible solution exists (and if so to find a BFS) as it is to find the optimal BFS given an initial BFS.....

checking feasibility via simplex

Consider the following new LP: **Maximize** $-x_0$

Subject to:

$$
\begin{array}{ll}
a_{1,1}\ x_1 + a_{1,2}\ x_2 + \ldots + a_{1,n}\ x_n - x_0 & \leq b_1 \\
a_{2,1}\ x_1 + a_{2,2}\ x_2 + \ldots + a_{2,n}\ x_n - x_0 & \leq b_2 \\
\ldots & \ldots \\
a_{m,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{m,n}\ x_n - x_0 & \leq b_m
\end{array}
$$

$$x_0, x_1, \ldots, x_n \geq 0$$

- This LP is feasible: let $x_0 = -\min\{b_1, \ldots, b_m, 0\}$, $x_j = 0$, for $j = 1, \ldots, n$. We can also get a feasible dictionary, and thus initial BFS, for it by adding slack variables.
- *Key point:* the original LP is feasible if and only if in an optimal solution to the new LP, $x_0^* = 0$.
- It also turns out, it is easy to derive a BFS for the original LP from an optimal BFS for this new LP.

how efficient is simplex?

- Each pivoting iteration can be performed in $O(mn)$ arithmetic operations.
  Also, it can be shown that the coefficients never get "too large" (they stay polynomial-sized), as long as rational coefficients are kept in reduced form (e.g., removing common factors from numerator and denominator).
  So, each pivot can be done in "polynomial time".
- How many pivots are required to get to the optimal solution? Unfortunately, it can be exponentially many!
- In fact, for most "pivoting rules" known, there exist worst case examples that force exponentially many iterations. (E.g., Klee-Minty (1972).)
- Fortunately, simplex tends to be very efficient in practice: requiring $O(m)$ pivots on typical examples.

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10].
  Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10].
  Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].
- Breakthrough: [Khachian'79] proved the LP problem does have a polynomial time algorithm, using a completely different approach: The "Ellipsoid Algorithm". Ellipsoid is theoretically very important, but not practical.

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10].
  Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].
- Breakthrough: [Khachian'79] proved the LP problem does have a polynomial time algorithm, using a completely different approach: The "Ellipsoid Algorithm". Ellipsoid is theoretically very important, but not practical.
- Breakthrough: [Karmarkar'84] gave a completely different P-time algorithm, using "the interior-point method". It is competitive with simplex in many cases.

final remarks and food for thought

- Why is the Simplex algorithm so fast in practice?
  Some explanation is offered by [Borgwardt'77]'s "average case" analysis of Simplex. More convincing explanation is offered by [Spielman-Teng'2001]'s "smoothed analysis" of Simplex (not "light reading").
- Ok. Enough about Simplex. So we now have an efficient algorithm for, among other things, finding minimax solutions to 2-player zero-sum games.
- Next time, we will learn about the very important concept of <u>Linear Programming Duality</u>.
  LP Duality is closely related to the Minimax theorem, but it has far reaching consequences in many subjects.
- **Food for thought:** Suppose you have a solution to an LP in Primal Form, and you want to convince someone it is optimal. How would you do it?