

CS675: Convex and Combinatorial Optimization
Fall 2014
The Simplex Algorithm

Instructor: Shaddin Dughmi

Algorithms for Convex Optimization

- We will look at 2 algorithms in detail: Simplex and Ellipsoid,
- Interior Point methods (e.g. gradient descent and variants) are important, in particular in practice, but we won't look at them in any depth

History and Basics of the Simplex Algorithm

- First methodical procedure for solving linear programs
- Developed by George Dantzig in 1947
- Considered one of the most influential algorithms of the 20th century

History and Basics of the Simplex Algorithm

- First methodical procedure for solving linear programs
- Developed by George Dantzig in 1947
- Considered one of the most influential algorithms of the 20th century
- Really a family of algorithms, parametrized by a “pivot rule”

History and Basics of the Simplex Algorithm

- First methodical procedure for solving linear programs
- Developed by George Dantzig in 1947
- Considered one of the most influential algorithms of the 20th century
- Really a family of algorithms, parametrized by a “pivot rule”
- Efficient in practice, leading to conjectures that it runs in polynomial time
- In 1972, Klee and Minty exhibited worst-case examples that take exponential time, at least for some of the most popular simplex pivot rules

History and Basics of the Simplex Algorithm

- First methodical procedure for solving linear programs
- Developed by George Dantzig in 1947
- Considered one of the most influential algorithms of the 20th century
- Really a family of algorithms, parametrized by a “pivot rule”
- Efficient in practice, leading to conjectures that it runs in polynomial time
- In 1972, Klee and Minty exhibited worst-case examples that take exponential time, at least for some of the most popular simplex pivot rules
- This spurred development of the Ellipsoid method, interior point methods, ...

Outline

- 1 Description of The Simplex Algorithm
- 2 Properties
- 3 Initialization

Linear Programming

We consider a standard form LP written as follows for convenience

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \preceq b \end{array}$$

- We use n to denote the number of variables, and m to denote the number of constraints.

Linear Programming

We consider a standard form LP written as follows for convenience

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \preceq b \end{array}$$

- We use n to denote the number of variables, and m to denote the number of constraints.
- Recall: optimal occurs at a vertex and corresponds to n linearly-independent tight inequalities

Linear Programming

We consider a standard form LP written as follows for convenience

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \preceq b \end{array}$$

- We use n to denote the number of variables, and m to denote the number of constraints.
- Recall: optimal occurs at a vertex and corresponds to n linearly-independent tight inequalities
- We assume we are given a starting vertex x_0 as input, and want to compute optimal vertex x^*
 - This is Phase II
 - Phase I, finding an initial vertex, involves solving another LP. We will come back to this at the end.

Linear Programming

We consider a standard form LP written as follows for convenience

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \preceq b \end{array}$$

- We use n to denote the number of variables, and m to denote the number of constraints.
- Recall: optimal occurs at a vertex and corresponds to n linearly-independent tight inequalities
- We assume we are given a starting vertex x_0 as input, and want to compute optimal vertex x^*
 - This is Phase II
 - Phase I, finding an initial vertex, involves solving another LP. We will come back to this at the end.
- Degeneracy: a vertex with $> n$ tight inequalities
 - We will mostly assume this away to save ourselves a headache

Linear Programming

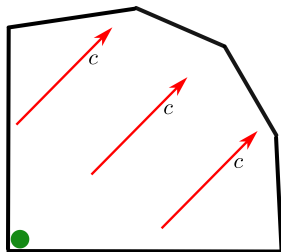
We consider a standard form LP written as follows for convenience

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \preceq b \end{array}$$

$$\begin{array}{ll} \text{minimize} & y^\top b \\ \text{subject to} & y^\top A = c^\top \\ & y \succeq 0 \end{array}$$

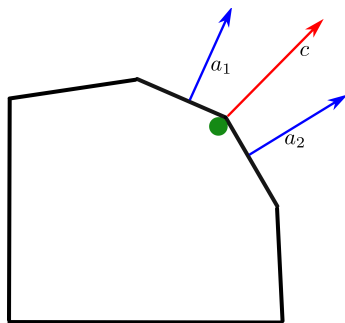
- We use n to denote the number of variables, and m to denote the number of constraints.
- Recall: optimal occurs at a vertex and corresponds to n linearly-independent tight inequalities
- We assume we are given a starting vertex x_0 as input, and want to compute optimal vertex x^*
 - This is Phase II
 - Phase I, finding an initial vertex, involves solving another LP. We will come back to this at the end.
- Degeneracy: a vertex with $> n$ tight inequalities
 - We will mostly assume this away to save ourselves a headache
- Incidentally, algorithm will produce optimal dual y^* as well.

Recall: Physical Interpretation of LP



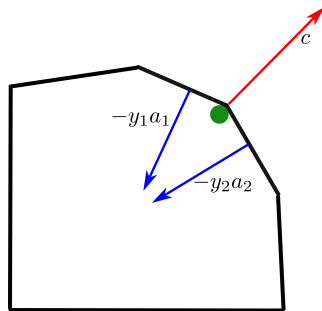
- Apply force field c to a ball inside bounded polytope $Ax \leq b$.

Recall: Physical Interpretation of LP



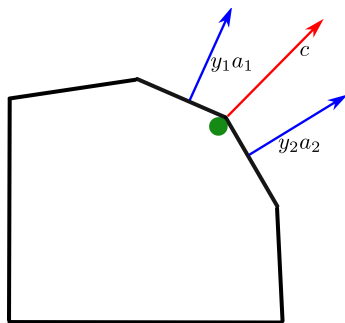
- Apply force field c to a ball inside bounded polytope $Ax \leq b$.
- Eventually, ball will come to rest against the walls of the polytope.

Recall: Physical Interpretation of LP



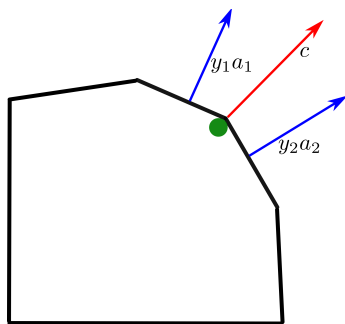
- Apply force field c to a ball inside bounded polytope $Ax \leq b$.
- Eventually, ball will come to rest against the walls of the polytope.
- Wall $a_i x \leq b_i$ applies some force $-y_i a_i$ to the ball for some $y_i \geq 0$

Recall: Physical Interpretation of LP



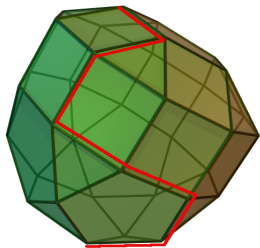
- Apply force field c to a ball inside bounded polytope $Ax \leq b$.
- Eventually, ball will come to rest against the walls of the polytope.
- Wall $a_i x \leq b_i$ applies some force $-y_i a_i$ to the ball for some $y_i \geq 0$
- Since the ball is still, $c^T = \sum_i y_i a_i = y^T A$.

Recall: Physical Interpretation of LP



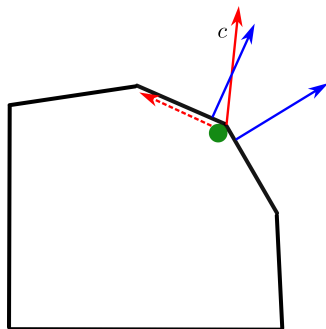
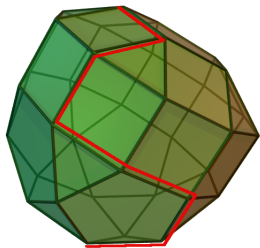
- Apply force field c to a ball inside bounded polytope $Ax \leq b$.
- Eventually, ball will come to rest against the walls of the polytope.
- Wall $a_i x \leq b_i$ applies some force $-y_i a_i$ to the ball for some $y_i \geq 0$
- Since the ball is still, $c^T = \sum_i y_i a_i = y^T A$.
- At optimality, only the walls adjacent to the ball push (Complementary Slackness)
 - Necessary and sufficient for optimality, given dual-feasible y

Informal Description



- Starts at initial vertex $x = x_0$
- While x is not optimal, move to a neighbouring vertex x' with $cx' > cx$.

Informal Description



- Starts at initial vertex $x = x_0$
- While x is not optimal, move to a neighbouring vertex x' with $cx' > cx$.
 - Either c is in the cone defined by tight constraints at x , in which case x is optimal by complementary slackness
 - Or else can improve cx by moving along an edge (1-d face)

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^\top = y^\top A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
- 2 If $y \geq 0$ then **stop and return** (x, y) , else
- 3 Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.
- 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
- 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^T = y^T A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
 - 2 If $y \geq 0$ then **stop and return** (x, y) , else
 - 3 Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.
 - 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
 - 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility
- Let T be set of tight rows. $y_T^T A_T = c^T$
 - Gaussian elimination

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^\top = y^\top A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
 - 2 If $y \geq 0$ then **stop and return** (x, y) , else
 - 3 Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.
 - 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
 - 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility
- y is a dual satisfying complementary slackness with x
 - Therefore both are optimal

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^\top = y^\top A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
 - 2 If $y \geq 0$ then **stop and return** (x, y) , else
 - 3 **Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.**
 - 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
 - 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility
- Chosen so that moving in direction d preserves tightness of $T \setminus \{i\}$, and loosens i .
 - A_T is full-rank, therefore $\text{null}(A_{T \setminus \{i\}})$ is a 1-dimensional subspace which is not normal to a_i
 - Choose $d \in \text{null}(A_{T \setminus \{i\}})$ appropriately.
 - Moving in direction d improves objective: $c^\top d = y^\top A d = y_i a_i d > 0$

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^\top = y^\top A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
 - 2 If $y \geq 0$ then **stop and return** (x, y) , else
 - 3 Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.
 - 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
 - 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility
- i.e. $Ad \leq 0$

Simplex Method

- **Input:** vertex $x = x_0$
- **Output:** Optimal vertex x^* and complementary dual y^* , or unbounded

Repeat the following:

- 1 Write $c^\top = y^\top A$, where $y_i \neq 0$ only for n tight constraints $a_i x = b_i$.
- 2 If $y \geq 0$ then **stop and return** (x, y) , else
- 3 Choose i with $y_i < 0$, and let \vec{d} be s.t. $A_{T \setminus \{i\}} d = 0$ and $a_i d = -1$.
- 4 If $x + \lambda d$ feasible for all $\lambda \geq 0$, **stop and return unbounded**, else
- 5 $x \leftarrow x + \lambda d$, for largest $\lambda \geq 0$ maintaining feasibility

- $\lambda = \min \left\{ \frac{b_j - a_j x}{a_j d} : j \in [m], a_j d > 0 \right\}$
- j achieving this minimum is a new tight constraint, replacing i .
- By nondegeneracy assumption, $\lambda > 0$

Outline

1 Description of The Simplex Algorithm

2 Properties

3 Initialization

Claim

If the simplex algorithm terminates, then it correctly outputs either an optimal primal/dual pair or unbounded.

- Primal feasibility of x is maintained throughout
- Returns (x, y) only if y is dual feasible and satisfies complementary slackness
 - x and y are both optimal
- Returns unbounded only if there is a direction d with $c^T d > 0$ and $Ad \leq 0$.

Termination in the Absence of Degeneracy

Claim

In the absence of degenerate vertices, the simplex algorithm terminates in a finite number of steps, at most $\binom{m}{n} \leq 2^m$.

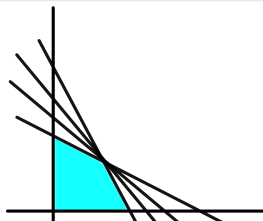
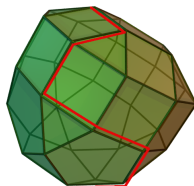
- There are at most $\binom{m}{n}$ distinct vertices in the polyhedron
- In the absence of degeneracy, the simplex algorithm does not repeat a vertex
 - In each iteration, moves along an edge in direction d , in total λd
 - We saw: $c^\top d > 0$, and $\lambda > 0$.
 - Objective strictly improves each iteration

Pivot Rules

Note

The algorithm we presented was not fully specified

- When multiple neighboring vertices are improving, which one should we choose so as to terminate as quickly as possible?
- In the presence of degeneracy, how should we identify the next (geometric) vertex so as to guarantee termination?
 - We maintain n tight and linearly independent constraints T , to be thought of as an algebraic representation of a vertex (aka a **basic feasible solution (BFS)**)
 - When many algebraic representations are possible of a single geometric vertex, unclear how to identify the next geometric vertex.



Pivot Rules

Both concerns are addressed by the use of a **pivot rule**, which determines the order in which we examine algebraic vertices.

Pivot rule

A rule for selecting which i leaves T , and which j enters T , when multiple choices are possible either because of multiple improving neighbors or degeneracy. Examples:

- Bland's rule: Choose lowest indexed i , then lowest indexed j
- Lexicographic: Maintain an order over rows, and move from T to the lexicographically smallest possible T' .
- Perturbation: perturb entries of b by a small value to remove degeneracy. This perturbation can be purely symbolic.

Runtime and Termination

- Many pivot rules, like the ones we mentioned, have been shown to never cycle over algebraic vertices
 - Guarantees termination in general, even in the presence of degeneracies
 - See book and notes for proofs.
- However, no pivot rules have been shown to guarantee a polynomial number of pivots
 - Even if no degeneracies.
- In 1972, Klee and Minty exhibited a family of examples that lead to exponential worst-case runtime for some widely-used pivot rules

Runtime and Termination

Nevertheless, one explanation as to the efficiency of the simplex algorithm in practice is through **smoothed complexity**

Theorem (Spielman & Teng '01)

*The simplex algorithm has polynomial **smoothed complexity**.*

- Model of input:
 - A, b, c chosen arbitrarily (worst case)
 - Then subjected to small gaussian noise with stddev σ (relative to largest entry of A, b, c)
 - Interpretation: measurement error
- More optimistic than worst case, but not quite as optimistic as average case.
- Expected runtime is polynomial in n, m and $\frac{1}{\sigma}$

Open Question

Is there a pivot rule which guarantees a polynomial number of pivots of the simplex algorithm in the worst case?

Why is this important?

- Would yield a **strongly** polynomial algorithm for LP
- If true, resolves in the affirmative a classic open question in polyhedral combinatorics
 - **Polynomial Hirsch Conjecture**: Is the diameter of the edge-vertex graph of an m -facet polytope in n -dimensional space bounded by a polynomial in n and m ?

Outline

1 Description of The Simplex Algorithm

2 Properties

3 Initialization

Solving a Linear Program via the Simplex Method

- Phase I: Find a vertex x_0 .
 - Phase II: Run the simplex algorithm starting from x_0
-
- So far, we have looked only at phase II
 - For phase I, we pose a different LP whose optimal solution is a vertex, if one exists

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

- If $x = 0$ is feasible, then it is a vertex and we are done, otherwise $b_{\min} < 0$

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & Ax - z\mathbf{1} \preceq b \\ & x \succeq 0 \\ & z \geq 0 \end{array}$$

- If $x = 0$ is feasible, then it is a vertex and we are done, otherwise $b_{\min} < 0$
- We write a new LP with a variable z measuring how far we are from feasibility

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & Ax - z\mathbf{1} \preceq b \\ & x \succeq 0 \\ & z \geq 0 \end{array}$$

- If $x = 0$ is feasible, then it is a vertex and we are done, otherwise $b_{\min} < 0$
- We write a new LP with a variable z measuring how far we are from feasibility
- If original LP is feasible, then an optimal solution new LP will have $z = 0$ and yield a feasible solution for original LP.

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & Ax - z\mathbf{1} \preceq b \\ & x \succeq 0 \\ & z \geq 0 \end{array}$$

- If $x = 0$ is feasible, then it is a vertex and we are done, otherwise $b_{\min} < 0$
- We write a new LP with a variable z measuring how far we are from feasibility
- If original LP is feasible, then an optimal solution new LP will have $z = 0$ and yield a feasible solution for original LP.
- An optimal vertex of new LP (with $z = 0$) will correspond to some vertex x_0 of original LP

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & Ax - z\mathbf{1} \preceq b \\ & x \succeq 0 \\ & z \geq 0 \end{array}$$

- We need a starting vertex for new LP, this is easier!
 - Let $x'_0 = 0$, and $z_0 = -b_{\min}$

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & Ax - z\vec{\mathbf{1}} \preceq b \\ & x \succeq 0 \\ & z \geq 0 \end{array}$$

- We need a starting vertex for new LP, this is easier!
 - Let $x'_0 = 0$, and $z_0 = -b_{\min}$
- Running simplex on new LP with starting vertex (x'_0, z_0) , we get starting vertex x_0 for original LP.