COMPLEXITY OF THE SIMPLEX ALGORITHM AND POLYNOMIAL-TIME ALGORITHMS

Béatrice Byukusenge

Linkping University

January 12, 2017
1. POLYNOMIAL COMPLEXITY ISSUES

2. COMPUTATIONAL COMPLEXITY OF THE SIMPLEX ALGORITHM

3. KARMARKAR’S PROJECTIVE ALGORITHM
Discuss fundamental computational complexity issues for algorithms for solving linear programming problems.

\( f(n) \) denotes "the total number of elementary operations required by the algorithm to solve the problem of size \( n \)."

\( f(n) = \mathcal{O}(n^k) \iff \exists \tau > 0 : f(n) \leq \tau n^k \) \( \text{Polynomial-time (theoretically efficient).} \)

\( f(n) = \mathcal{O}(k^n) \iff \exists \tau > 0 : f(n) \leq \tau k^n \) \( \text{exponential growth (bad!). e.g.: simplex algorithm.} \)

There exist theoretically efficient algorithms for LP problems:

- Khachian (no practical value).
- Karmarkar (promising).
Consider the LP optimization problem:

\[
\begin{align*}
\text{minimize} \quad & z(x) = cx \\
\text{s. to} \quad & Ax = b \\
& \mathbb{R}^n \ni x \geq 0
\end{align*}
\]

Data: \( A \in \mathbb{R}^{m \times n}; \ c \in \mathbb{R}^n; \ b \in \mathbb{R}^m \) with \( m, n \geq 2 \).

- **size:** \((m, n, L)\), where \( L \) is the **input length:** the number of binary bits required to record all the data of the problem (here \( \log = \log_2 \)):

\[
L = \left\{1 + \lceil \log(1 + m) \rceil\right\} + \left\{1 + \lceil \log(1 + n) \rceil\right\} \\
+ \sum_j \left\{1 + \lceil \log(1 + |c_j|) \rceil\right\} + \sum_i \sum_j \left\{1 + \lceil \log(1 + |a_{ij}|) \rceil\right\} \\
+ \sum_i \left\{1 + \lceil \log(1 + |b_i|) \rceil\right\}.
\]
We are only required to determine a function $g(m, n, L)$ in terms of $(m, n, L)$ such that for some sufficiently large constant $\tau > 0$, we have

- $f(n, m, L) \leq \tau g(m, n, L)$. i.e., $O(g(m, n, L))$.

**Example:** For algorithm actually involving a maximum of $f(n, m) = 6m^2n + 15mn + 12m$ is $O(m^2, n)$. 
We are only required to determine a function $g(m, n, L)$ in terms of $(m, n, L)$ such that for some sufficiently large constant $\tau > 0$, we have

\[ f(n, m, L) \leq \tau g(m, n, L). \text{ i.e., } \mathcal{O}(g(m, n, L)). \]

**Example:** For algorithm actually involving a maximum of $f(n, m) = 6m^2n + 15mn + 12m$ is $\mathcal{O}(m^2, n)$.

---

**Optimization Problem**

maximize \[ z(x) = cx \]

s. to \[ Ax \leq b \]

\[ x \geq 0 \]

**Decision Problem**

Given \( c, b \) and \( A \) (of the appropriate dimensions) and given rational number \( K \), does there exist a rational vector \( x \) such that \( Ax = b \), \( x \geq 0 \), and \( cx \leq K \)?
We are only required to determine a function \( g(m, n, L) \) in terms of \((m, n, L)\) such that for some sufficiently large constant \( \tau > 0 \), we have
\[
 f(n, m, L) \leq \tau g(m, n, L). \text{ i.e., } \mathcal{O}(g(m, n, L)).
\]

**Example:** For algorithm actually involving a maximum of \( f(n, m) = 6m^2n + 15mn + 12m \) is \( \mathcal{O}(m^2, n) \).

---

**Optimization Problem**

maximize \( z(x) = cx \)

s. to \( Ax \leq b \)

\( x \geq 0 \)

---

**Decision Problem**

Given \( c, b \) and \( A \) (of the appropriate dimensions) and given rational number \( K \), does there exist a rational vector \( x \) such that \( Ax = b \), \( x \geq 0 \), and \( cx \leq K \)?

---

**Theorem**

*polynomial-time algorithms for optimization problems* \( \Leftrightarrow \) *those for decision problems.*

Béatrice Byukusenge
Dantzig introduces the **simplex algorithm**.

- **intuition-based reaction:** the algorithm would not prove to be very efficient.

- **surprisingly:** in practice, this method performes exceedingly well.

Theoretically, the fact is that the algorithm is entrapped in the potentially combinatorial aspect of having to examine up to (for \( n > m \)):

\[
\binom{n}{m} > \left( \frac{n}{m} \right)^m \text{ vertices.}
\]

- Hence the plausibility of a potential **exponential order of effort for some problems**.
Example: 1971 Klee-Minty problems: Feasible region is a suitable distortion of the n-dimensional hypercube in \( \mathbb{R}^n \) which has \( 2^n \) vertices.

### Problem (\( \varepsilon \in (0, 1/2) \))

Maximize \( x_n \)

s. to \( 0 \leq x_1 \leq 1 \)

\[ \varepsilon x_{j-1} \leq x_j \leq 1 - \varepsilon x_{j-1} \]  
(for \( j = 2, \ldots, n \))

\( x_j \geq 0, \ j = 1, \ldots, n. \)

---

### Transformed Problem (\( \theta = 1/\varepsilon \))

Maximize \( \sum_{j=1}^{n} y_j \)

s. to \( y_1 \leq 1 \)

\[ y_j + 2 \sum_{k=1}^{j-1} y_k \leq \theta^{j-1} \]  
(for \( j = 2, \ldots, n \))

\( y_j \geq 0, \ j = 1, \ldots, n. \)

where \( y_1 = x_1, \ y_j = (x_j - \varepsilon x_{j-1})/\varepsilon^{j-1} \) for \( j = 2, \ldots, n. \)

\( 2^n - 1 \) iterations to visit all the \( 2^n \) vertices.
Figure 8.1. Illustration of the Klee–Minty type polytopes for $n=2$ and $n=3$. 

(a) Case for $n = 2$

(b) Case for $n = 3$
In 1984 Karmarkar (AT&T Bell Laboratories) proposed a new **polynomial-time** algorithm for LP problems. This algorithm addresses LP problems of the following form:

Minimize \( z = cx \)

s. to \( Ax = 0 \)

\( 1x = 1 \) \hspace{1cm} (LP-K)

\( x \geq 0 \)

where \( A \in \mathbb{R}^{m \times n} \), with \( m, n \geq 2 \), \( c, A \) integers and \( 1 \) is a row vector of \( n \) ones with the following two assumptions:

- \((A_1)\): \( x_0 = (\frac{1}{n}, \ldots, \frac{1}{n})^T \) is feasible.
- \((A_2)\): \( z^* = 0 \).
Any general LP problem can be \((\text{polynomially})\) cast in this form through the use of artificial variables, an artificial bounding constraint, and through variable redefinitions.

- **Remark:** Under assumptions \((A_1)\) and \((A_1)\), Problem \((LP - K)\) is feasible and bounded, and hence, has an optimum.
Feasible region: \( K = \{Ax = 0\} \cap \{S_x \{x : 1x = 1, x \geq 0\}\} \)
Summary of Karmarkar’s Algorithm

**INITIALIZATION**

Compute \( r = 1/\sqrt{n(n-1)} \), \( L = \left[ 1 + \log \left( 1 + |c_{j_{\text{max}}}| \right) + \log \left( |\text{det}_{\text{max}}| \right) \right] \), and select \( \alpha = (n-1)/3n \). Let \( x_0 = (1/n, \ldots, 1/n)^t \) and put \( k = 0 \).
If $cx_k < 2^{-L}$, use the optimal rounding routine to determine an optimal solution, and stop. (Practically, since $2^{-L}$ may be very small, one may terminate when $cx_k$ is less than some other desired tolerance.) Otherwise, define

$$D_k = \text{diag}\{x_{k1}, \ldots, x_{kn}\}, \quad y_0 = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)^t,$$

$$P = \left[ \begin{array}{c} ADk \\ 1 \end{array} \right] \quad \text{and} \quad \bar{c} = cD_k$$

and compute

$$y_{\text{new}} = y_0 - \alpha r \frac{c_p}{\|c_p\|}, \quad \text{where} \quad c_p = \left[ I - P^t (PP^t)^{-1} P \right] \bar{c}^t.$$

Hence, obtain $x_{k+1} = (D_k y_{\text{new}}) / (1D_k y_{\text{new}})$. Increment $k$ by one and repeat the Main Step.
OPTIMAL ROUNDING ROUTINE

Starting with $x_k$, determine an extreme point solution $\bar{x}$ for Problem (8.4) with $c\bar{x} \leq cx_k < 2^{-L}$, using the earlier purification scheme. Terminate with $\bar{x}$ as an optimal solution to Problem (8.4).
Thank you for your attention!