

# The Best of 20<sup>th</sup> Century: Top 10 Algorithms

from *SIAM News*, Volume 33, Number 4

## The Best of the 20th Century: Editors Name Top 10 Algorithms

By Barry A. Cipra

*Algos* is the Greek word for pain. *Algor* is Latin, to be cold. Neither is the root for *algorithm*, which stems instead from al-Khwarizmi, the name of the ninth-century Arab scholar whose book *al-jabr wa'l muqabalah* devolved into today's high school algebra textbooks. Al-Khwarizmi stressed the importance of methodical procedures for solving problems. Were he around today, he'd no doubt be impressed by the advances in his eponymous approach.

Some of the very best algorithms of the computer age are highlighted in the January/February 2000 issue of *Computing in Science & Engineering*, a joint publication of the American Institute of Physics and the IEEE Computer Society. Guest editors Jack Dongarra of the University of Tennessee and Oak Ridge National Laboratory and Francis Sullivan of the Center for Computing Sciences at the Institute for Defense Analyses put together a list they call the "Top Ten Algorithms of the Century."

"We tried to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century," Dongarra and Sullivan write. As with any top-10 list, their selections—and non-selections—are bound to be controversial, they acknowledge. When it comes to picking the algorithmic best, there seems to be no best algorithm.

Without further ado, here's the CiSE top-10 list, in chronological order. (Dates and names associated with the algorithms should be read as first-order approximations. Most algorithms take shape over time, with many contributors.)

# The Best of 20<sup>th</sup> Century: Top 10 Algorithms

**1946:** John von Neumann, Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the Metropolis algorithm, also known as the **Monte Carlo method**.

The Metropolis algorithm aims to obtain approximate solutions to numerical problems with unmanageably many degrees of freedom and to combinatorial problems of factorial size, by mimicking a random process. Given the digital computer's reputation for deterministic calculation, it's fitting that one of its earliest applications was the generation of random numbers.



*In terms of widespread use, George Dantzig's simplex method is among the most successful algorithms of all time.*

**1947:** George Dantzig, at the RAND Corporation, creates the **simplex method for linear programming**.

In terms of widespread application, Dantzig's algorithm is one of the most successful of all time: Linear programming dominates the world of industry, where economic survival depends on the ability to optimize within budgetary and other constraints. (Of course, the "real" problems of industry are often nonlinear; the use of linear programming is sometimes dictated by the computational budget.) The simplex method is an elegant way of arriving at optimal answers. Although theoretically susceptible to exponential delays, the algorithm in practice is highly efficient—which in itself says something interesting about the nature of computation.

**1950:** Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of **Krylov subspace iteration methods**.

These algorithms address the seemingly simple task of solving equations of the form  $Ax = b$ . The catch, of course, is that  $A$  is a huge  $n \times n$  matrix, so that the algebraic answer  $x = b/A$  is not so easy to compute. (Indeed, matrix "division" is not a particularly useful concept.) Iterative methods—such as solving equations of the form  $Kx_{i+1} = Kx_i + b - Ax_i$  with a simpler matrix  $K$  that's ideally "close" to  $A$ —lead to the study of Krylov subspaces. Named for the Russian mathematician Nikolai Krylov, Krylov subspaces are spanned by powers of a matrix applied to an initial "remainder" vector  $r_0 = b - Ax_0$ . Lanczos found a nifty way to generate an orthogonal basis for such a subspace when the matrix is symmetric. Hestenes and Stiefel proposed an even niftier method, known as the conjugate gradient method, for systems that are both symmetric and positive definite. Over the last 50 years, numerous researchers have improved and extended these algorithms. The current suite includes techniques for non-symmetric systems, with acronyms like GMRES and Bi-CGSTAB. (GMRES and Bi-CGSTAB premiered in *SIAM Journal on Scientific and Statistical Computing*, in 1986 and 1992,

# The Best of 20<sup>th</sup> Century: Top 10 Algorithms

- 1946: Monte Carlo method
- **1947: Simplex method for linear programming (Dantzig)**
- 1950: Krylov subspace iteration method
- 1951: Decompositional approach to matrix computation
- 1957: Fortran optimizing compiler
- 1959–61: computing eigenvalues, QR algorithm
- 1962: Quicksort
- 1965: Fast Fourier Transformation
- 1977: Integer relation detection algorithm
- 1987: Fast multipole algorithm

# Линеарно “програмирање”

- Линеарно програмирање је специјалан случај нелинеарног програмирања (NLP)
- Оптимизациона функција је линеарна функција

$$f(x_1, x_2, \dots, x_D) = c_1 x_1 + c_2 x_2 + \dots + c_D x_D = \sum_{k=1}^D c_k x_k$$

- Услови су линеарне функције (једнакости или неједнакости)

$$a_1 x_1 + a_2 x_2 + \dots + a_D x_D \begin{cases} \leq \\ = \\ \geq \end{cases} b_1$$

# Формализација LP (стандардни облик)

- Максимизације оптимизационе функције

$$f(x_1, x_2, \dots, x_D) = c_1 x_1 + c_2 x_2 + \dots + c_D x_D = \sum_{k=1}^D c_k x_k$$

- Услови (укупно  $m$  услова,  $b_1, b_2, \dots, b_m \geq 0$ )

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1D}x_D \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2D}x_D \leq b_2$$

⋮

,

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mD}x_D \leq b_m$$

$$x_1, x_2, \dots, x_D \geq 0$$

# Превођење у стандардни облик

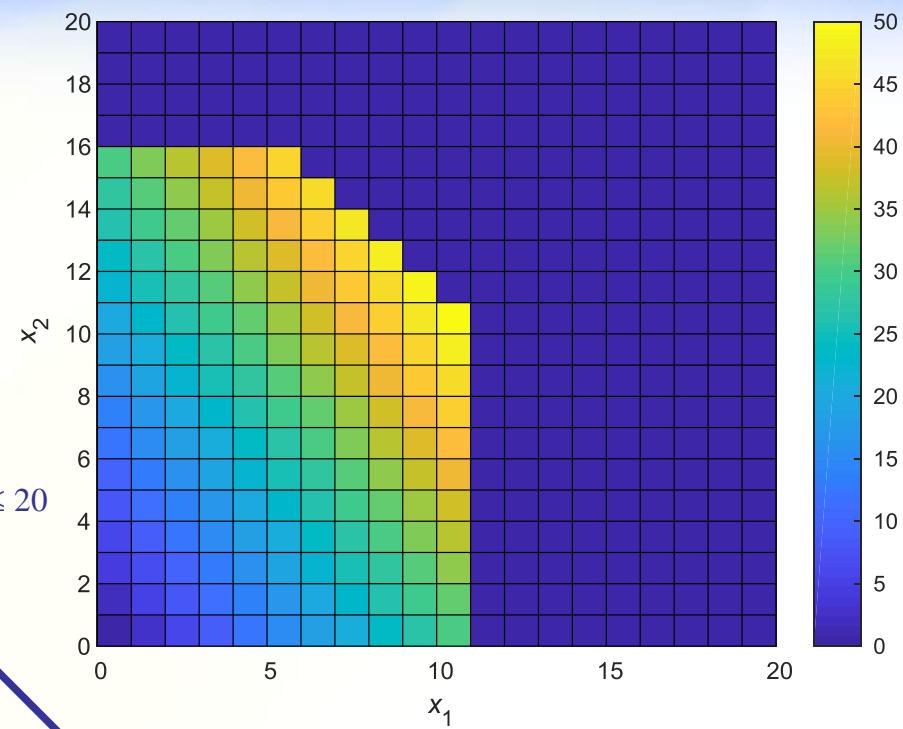
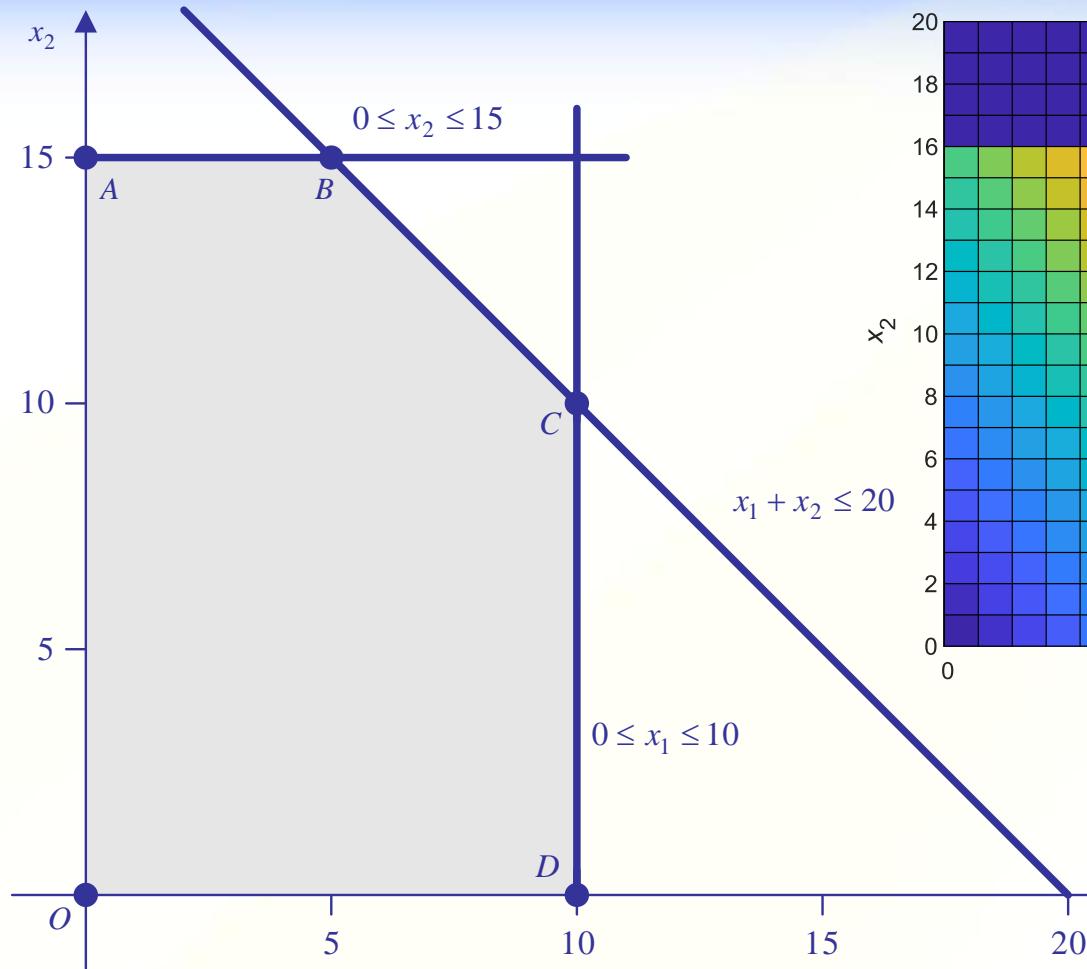
- $\min f \rightarrow \max (-f)$
- Све променљиве морају бити ненегативне
  - неограничена променљива  
 $x \rightarrow x = x_1 - x_2$  при чему је  $x_1, x_2 \geq 0$
  - непозитивна променљива:  
 $x \leq 0 \rightarrow x = x_1 - x_2$  при чему је  $x_1, x_2 \geq 0$   
(најједноставније  $x_1=0$  и  $x_2 \geq 0$ )
- Једнакости
  - Редукција броја непознатих  
 $ax_1 + bx_2 = c \rightarrow x_1 = (c - bx_2)/a$  и замена у свим изразима где се појављује  
(може да буде компликовано у случају великог броја услова) или
  - Претварање у две неједнакости  
 $ax_1 + bx_2 = c \rightarrow ax_1 + bx_2 \leq c$  и  $ax_1 + bx_2 \geq c$
- Све неједнакости се преводе у облик  
 $ax_1 + bx_2 \leq c$

# Пример: проблем интернет провайдера

Интернет провайдер са ограниченим пропусним опсегом. Две опције за корисника (1) неограничен приступ (flat-rate) и (2) приступ са ограниченим протоком и количином података. Нека је број корисника са неограниченом приступом означен са  $x_1$ , а број корисника са ограниченим приступом  $x_2$ . Бројеви корисника припадају скупу целих бројева. Корисници који приступају интернету са неограниченом приступом плаћају месечно  $\alpha$ , а корисници који приступају интернету са ограниченим приступом плаћају  $\beta$ , при чему је  $\alpha > \beta$ . Услед ограничених капацитета провайдера, постоје горње границе за  $x_1$  и  $x_2$ , као и за  $x_1 + x_2$ . Циљ провайдера је да максимизира зараду  $f(x_1, x_2) = \alpha x_1 + \beta x_2$ . Који је оптималан избор  $x_1$  и  $x_2$ ?

Ради илустрације, сматраћемо да је  $\alpha = 3$ ,  $\beta = 2$ ,  $0 \leq x_1 \leq 10$ ,  $0 \leq x_2 \leq 15$  и  $x_1 + x_2 \leq 20$ .

# Графички приступ



# Домен решења и екстремне тачке

- Домен решења је означен сивом бојом
- Домен је ограничен тачкама  $O, A, B, C, D$  које представљају крајње (екстремне) тачке
- Колико има екстремних тачака у општем случају са  $N$  услова?
- Решење проблема мора бити у једној од екстремних тачака домена решења

# Решење проблема

- Како решити проблем у општем случају?
- Одређивање свих екстремних тачака домена и провера описне функције у њима

$$f_O(0,0) = 0$$

$$f_A(0,15) = 30$$

$$f_B(5,15) = 45$$

$$f_C(10,10) = 50$$

$$f_D(10,0) = 30$$

# Dantzig simplex алгоритам

- Најпознатији алгоритам за решавање LP
- G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, 1959.
- George B. Dantzig and Mukund N. Thapa. 1997. *Linear programming 1: Introduction*. Springer-Verlag.
- George B. Dantzig and Mukund N. Thapa. 2003. *Linear Programming 2: Theory and Extensions*. Springer-Verlag.
- 1975. Нобелова награда за LP програмирање у економији L. Kantorovich и T.C. Koopmans

# Почетак алгоритма

- Припрема за примену алгоритма, превођење неједнакости у једнакости  
увођење помоћне (изравнавајуће) променљиве (енг: slack variable)

$$x_1 + x_2 \leq 20 \rightarrow x_1 + x_2 + s_1 = 20, \quad s_1 \geq 0$$

$$x_1 \leq 10 \rightarrow x_1 + s_2 = 10, \quad s_2 \geq 0$$

$$x_2 \leq 15 \rightarrow x_2 + s_3 = 15, \quad s_3 \geq 0$$

$$x_1, x_2 \geq 0$$

$$s_1, s_2, s_3 \geq 0$$

- Све неједнакости претворене у једнакости,  
**све променљиве су ненегативне**
- Могуће је добити
  - поддетерминисан систем линеарних једначина
  - систем линеарних једначина
  - предетерминисан систем линеарних једначина

# Компактан (матрични) запис услови и опт. функције

- Услови

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 15 \end{bmatrix} \rightarrow \mathbf{Ax} = \mathbf{b}$$

- Оптимизациона функција (max)

$$f(\mathbf{x}) = Z = \sum_{k=1}^D c_k x_k = \mathbf{c}^T \mathbf{x} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

# Канонични облик

- Циљ: максимизирати  $Z$
- Канонични облик LP (simplex tableau)

$$\begin{bmatrix} 1 & -\mathbf{c}^T \\ 0 & \mathbf{A} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$$

- У општем случају може се свести на овај облик  
(све једнакости и све променљиве ненегативне)

# Dantzig simplex

- Поћи из једне екстремне тачке  
(подразумева се да је бар једна таква тачка позната, што у општем случају не мора да буде тривијално)
- Проверити да ли је могуће повећати  $Z$  преласком у другу суседну екстремну тачку
- Завршити алгоритам уколико није могуће повећати  $Z$  (решење је пронађено)

# Dantzig алгоритам за проблем интернет провајдера: почетак

- $\max f = 3x_1 + 2x_2 = Z$   
 $x_1 + x_2 + s_1 = 20$   
 $x_1 + s_2 = 10$   
 $x_2 + s_3 = 15$
- зависне променљиве:  
 $x_1, x_2 \geq 0$
- независне (основне) променљиве:  
 $s_1, s_2, s_3 \geq 0$
- Одређивање полазне тачке:  
све зависне променљиве су 0  
 $x_1 = x_2 = 0$  следи  $Z = 0$

# Dantzig алгоритам за проблем интернет провајдера

Полазимо из тачке  $(x_1, x_2) = (0,0)$ , на слици је то тачка  $O$ , а  $Z = 0$ .

Основне променљиве су оне које чине јединичну субматрицу,  $s_1, s_2, s_3$ .

$$\begin{bmatrix} 1 & -3 & -2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 20 \\ 10 \\ 15 \end{bmatrix}$$

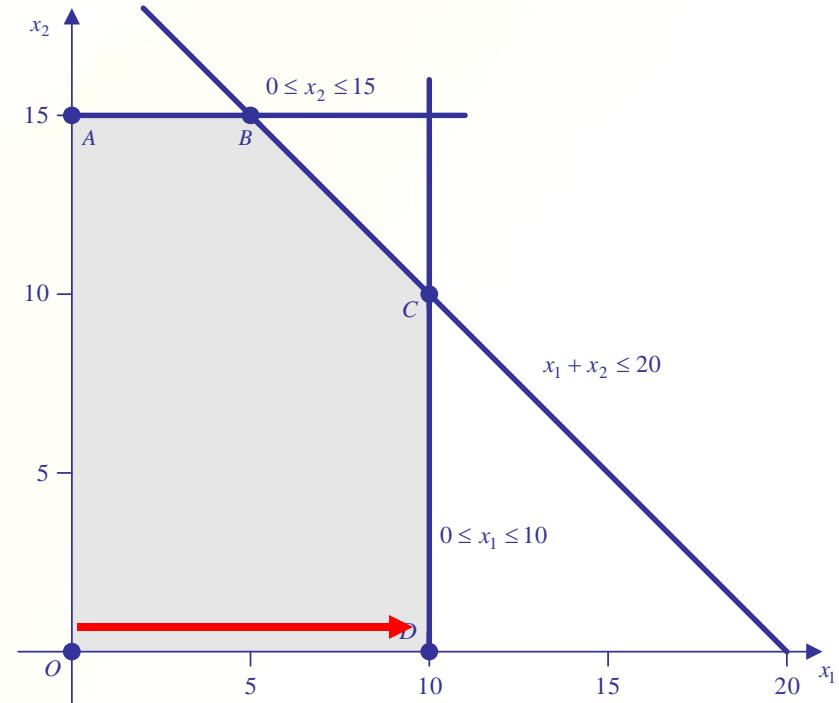
Основне променљиве (оне које чине јединичну субматрицу)  
називају се базисни вектор

# Пивотизација (основни корак алгоритма)

- Пронаћи зависну променљиву са позитивним коефицијентом у  $Z$  (повећање те променљиве повећава  $Z$ )
- Повећати променљиву максимално могуће на основу услова који постоје
- Пронаћи најстрожи услов (најмање повећање)
- Заменити основну и зависну променљиву

# Испитивање за први корак

- $Z = 3x_1 + 2x_2$   
 $x_1 + x_2 + s_1 = 20$   
 $x_1 + s_2 = 10$   
 $x_2 + s_3 = 15$
- $x_1$  има позитивни коефицијент  
 $x_1 + 0 + 0 = 20 \rightarrow x_{1\max} = 20$   
 $x_1 + 0 = 10 \rightarrow x_{1\max} = 10$   
 $x_2 + s_3 = 15$
- Најстрожи услов  $x_{1\max} = 10$
- $Z = 3(10 - s_2) + 2x_2 = 30 - 3s_2 + 2x_2$   
 $- s_2 + x_2 + s_1 = 10$   
 $x_1 = 10 - s_2$   
 $x_2 + s_3 = 15$
- Ново решење  $(x_1, x_2) = (10, 0)$ ,  $Z_{\max} = 30$



# Еквивалентне матричне манипулације

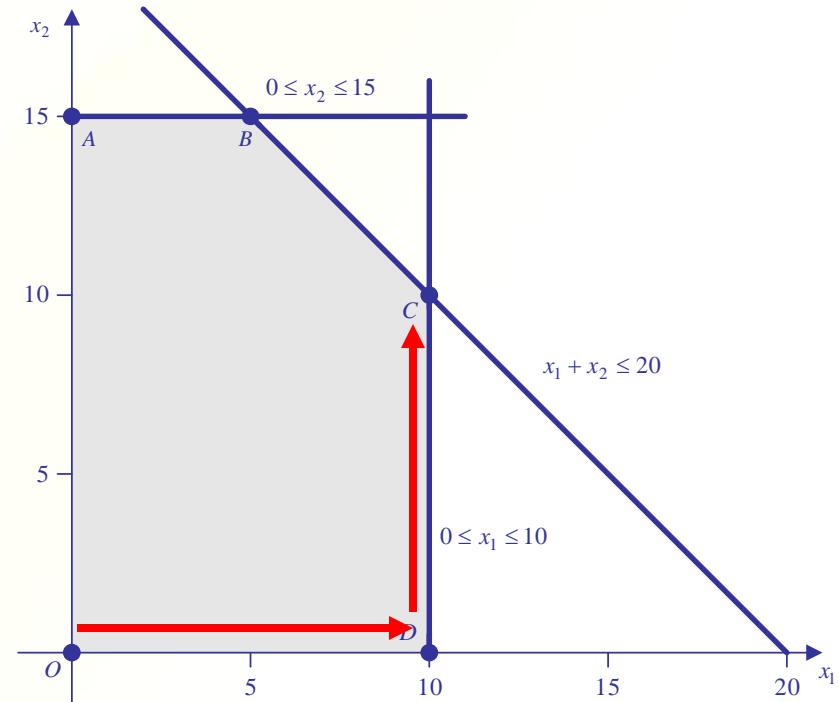
$$\begin{array}{c}
 \xrightarrow[3]{+} \\
 \left[ \begin{array}{cccccc} 1 & -3 & -2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \left[ \begin{array}{c} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{array} \right] = \left[ \begin{array}{c} 0 \\ 20 \\ 10 \\ 15 \end{array} \right] \rightarrow \left[ \begin{array}{cccccc} 1 & 0 & -2 & 0 & 3 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \left[ \begin{array}{c} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{array} \right] = \left[ \begin{array}{c} 30 \\ 20 \\ 10 \\ 15 \end{array} \right] \xleftarrow[-1]{+} \\
 \end{array}$$

$$\rightarrow \left[ \begin{array}{cccccc} 1 & 0 & -2 & 0 & 3 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \left[ \begin{array}{c} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{array} \right] = \left[ \begin{array}{c} 30 \\ 10 \\ 10 \\ 15 \end{array} \right]$$

- Табло се типично пише без вектора  $[Z \ x \ s]^T$  и знака једнакости
- Они су овде написан ради јасног сагледавања везе са полазним системом

# Испитивање за наредни корак

- $Z = 3(10-s_2) + 2x_2 = 30 - 3s_2 + 2x_2$   
 $-s_2 + x_2 + s_1 = 10$   
 $x_1 = 10 - s_2$   
 $x_2 + s_3 = 15$
- $x_2$  има позитивни коефицијент  
 $-0 + x_2 + 0 = 10 \rightarrow x_{2\max} = 10$   
 $x_1 = 10 - s_2$   
 $x_2 + s_3 = 15 \rightarrow x_{2\max} = 15$   
Најстрожи услов  $x_{2\max} = 10$
- $Z = 50 - 2s_1 - s_2$   
 $x_2 = 10 + s_2 - s_1$   
 $x_1 = 10 - s_2$   
 $-s_1 + s_2 + s_3 = 5$
- Ново решење  $(x_1, x_2) = (10, 10)$ ,  $Z_{\max} = 50$



# Еквивалентне матричне манипулације

$$\begin{array}{c}
 \text{+} \\
 \text{2} \times \text{C}_1 \rightarrow \\
 \left[ \begin{array}{cccccc} 1 & 0 & -2 & 0 & 3 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \begin{bmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 30 \\ 10 \\ 10 \\ 15 \end{bmatrix} \rightarrow \begin{array}{l} \left[ \begin{array}{cccccc} 1 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \begin{bmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 50 \\ 10 \\ 10 \\ 15 \end{bmatrix} \\ -1 \times \text{C}_2 \\
 \end{array}
 \end{array}$$

$$\rightarrow \left[ \begin{array}{cccccc} 1 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 \end{array} \right] \begin{bmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 50 \\ 10 \\ 10 \\ 5 \end{bmatrix}$$

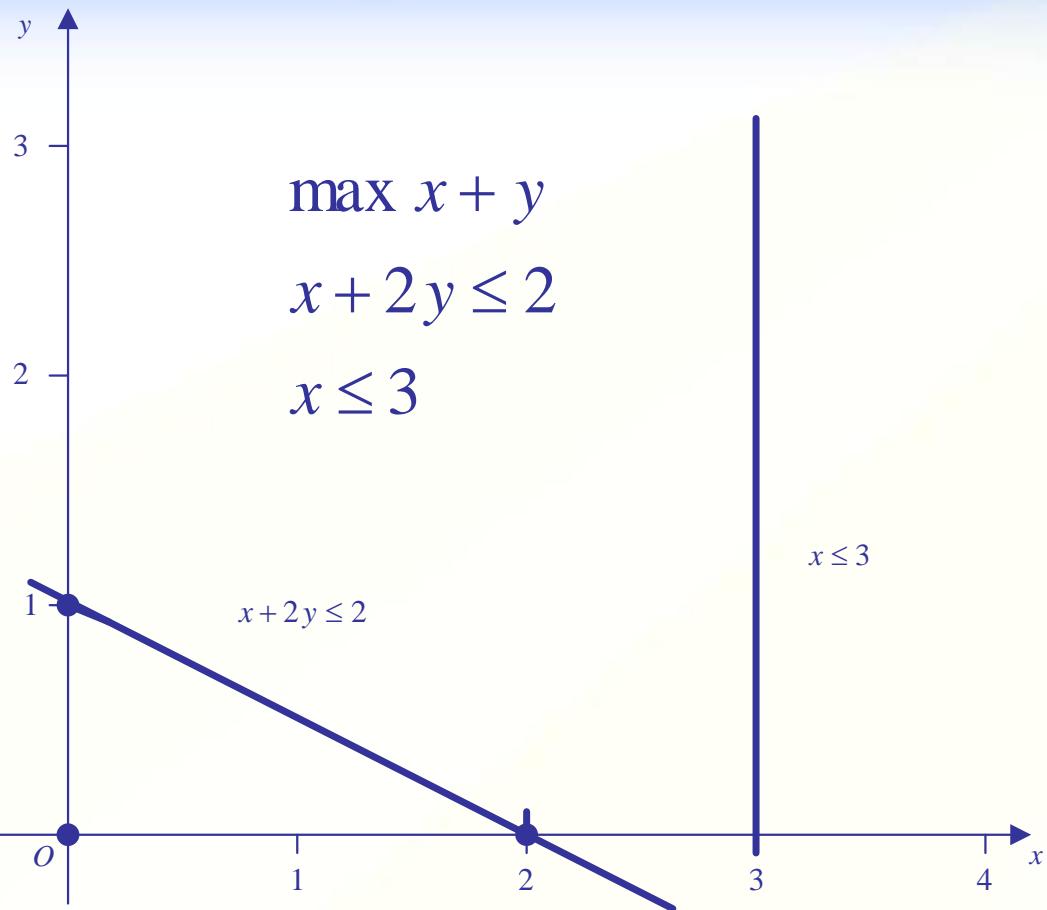
# Анализа наредног корака

- $Z = 50 - 2s_1 - s_2$
- Нема више променљивих које имају позитивни коефицијент у  $Z$
- Нема могућности за даље повећање  $Z$
- Решење је пронађено
- Крај алгоритма

# Генерализација

- Уколико су чланови са десне стране уговора  $b_i$  ненегативни, постоји решење
- Како организовати алгоритам уколико су  $b_i$  **негативни**?
- Описани поступак назива се “фаза 2” симплекс алгоритма
- Претпроцесирање (“фаза 1”) своди све друге LP проблеме у облик који се може решити “фазом 2”
- Потребно је пронаћи базисни вектор
  - додају се “вештачке променљиве”
  - pivotизацијом се решава “додатни проблем” који даје формулацију полезног проблема који се може решити “фазом 1”
- Може се догодити да полазни проблем
  - нема решење
  - оптимизациона функција (max) није ограничена са горње стране
  - ...
- Претпроцесирање је сложено у општем случају

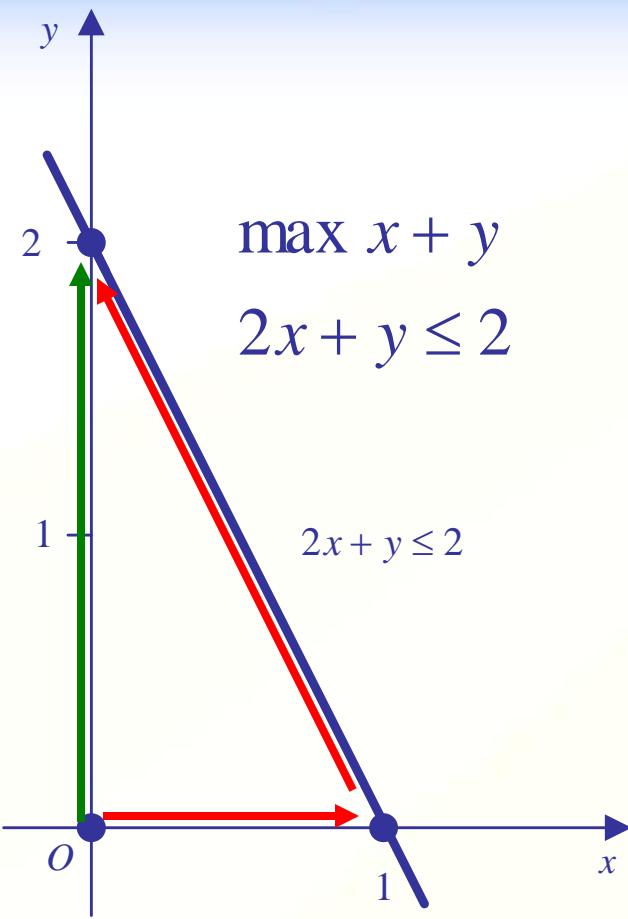
# Генерализација: Сувишни услов



$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 & 1 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 2 \\ 0 & 1 & 2 & 1 & 0 & 2 \\ 0 & 0 & -2 & -1 & 1 & 1 \end{bmatrix}$$

- Појављује се негативни коефицијент
- Прескочити такав услов

# Ефикасност: Избор наредног корака



$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 2 & 1 & 1 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & -0,5 & 0,5 & 1 \\ 0 & 2 & 1 & 1 & 2 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 2 & 1 & 1 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 1 & 2 \end{bmatrix}$$

- У првом случају **два корака** у другом случају **један корак**
- Од избора наредне тачке зависи ефикасност

# Транспортни проблем (услови су искључиво једнакости)

- Компанија производи штампаче у две фабрике F1 и F2 на различитим местима.
- Поред тога, има три продајна објекта C1, C2 и C3.
- Месечни захтеви објеката (у пакетима од по 10 штампача) су 8, 5 и 2.
- Фабрике F1 и F2 могу да произведу 6 и 9 пакета месечно, редом.
- Преносе се искључиво пакети (није могуће дељење пакета на мање јединице).
- Приметити да је укупна производња једнака укупној потражњи.
- Транспорт из фабрика до продајних објеката има различиту цену због различитих места на којима се они налазе.
- Цене транспорта су дате у табели и изражене су у хиљадама динара.
- Пронаћи план испоруке (колико пакета иде у који продајни центар и из које фабрике) тако да трошкови транспорта буду минимални.
- Израчунати минималне трошкове транспорта у том случају.

Цене преноса из Fx у Cy

	C1	C2	C3
F1	5	5	3
F2	6	4	1

# Формулација

- Трошкови испоруке су

$$z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23}$$

где је  $x_{ij}$  број пакета који иде из фабрике  $i$  у објекат  $j$  ( $i \in \{1,2\}$  и  $j \in \{1,2,3\}$ )

- Потребно је поронаћи све  $x_{ij}$  тако да функција  $z$  има минималну вредност, при задатим условима
- $x_{ij}$  мора бити цео број и ненегативан ( $x_{ij} \geq 0$ )

# Формулација са условима

- $\min z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23}$
- Услови
  - $x_{11} + x_{21} = 8$
  - $x_{12} + x_{22} = 5$
  - $x_{13} + x_{23} = 2$
  - $x_{11} + x_{12} + x_{13} = 6$
  - $x_{21} + x_{22} + x_{23} = 9$
- Сви услови су једнакости!  
Нема помоћних променљивих

# Једноставно решење елиминацијом променљивих

- Сваку једнакост можемо искористити да елиминишемо по једну променљиву

Елиминација  $x_{11}$

$$\min z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23} \Rightarrow z = 40 + 5x_{12} + 3x_{13} + x_{21} + 4x_{22} + x_{23}$$

$$x_{11} + x_{21} = 8 \Rightarrow x_{11} = 8 - x_{21}$$

$$x_{12} + x_{22} = 5$$

$$x_{13} + x_{23} = 2$$

$$x_{11} + x_{12} + x_{13} = 6 \Rightarrow -x_{21} + x_{12} + x_{13} = -2$$

$$x_{21} + x_{22} + x_{23} = 9$$

# Наредни кораци елиминације ( $x_{12}$ )

- Жуто означене једначине су већ искоришћене

Елиминација  $x_{12}$

$$z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23} \Rightarrow z = 65 + 3x_{13} + x_{21} - x_{22} + x_{23}$$

$$x_{11} + x_{21} = 8 \Rightarrow x_{11} = 8 - x_{21}$$

$$x_{12} + x_{22} = 5 \Rightarrow x_{12} = 5 - x_{22}$$

$$x_{13} + x_{23} = 2$$

$$x_{11} + x_{12} + x_{13} = 6 \Rightarrow -x_{21} + x_{12} + x_{13} = -2 \Rightarrow -x_{21} - x_{22} + x_{13} = -7$$

$$x_{21} + x_{22} + x_{23} = 9$$

# Наредни кораци елиминације ( $x_{13}$ )

- Жуто означене једначине су већ искоришћене

Елиминација  $x_{13}$

$$z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23} \Rightarrow z = 71 + x_{21} - x_{22} - 2x_{23}$$

$$x_{11} + x_{21} = 8 \Rightarrow x_{11} = 8 - x_{21}$$

$$x_{12} + x_{22} = 5 \Rightarrow x_{12} = 5 - x_{22}$$

$$x_{13} + x_{23} = 2 \Rightarrow x_{13} = 2 - x_{23}$$

$$x_{11} + x_{12} + x_{13} = 6 \Rightarrow -x_{21} + x_{12} + x_{13} = -2 \Rightarrow -x_{21} - x_{22} - x_{23} = -9$$

$$x_{21} + x_{22} + x_{23} = 9$$

Последње две једначине су идентичне!

# Наредни кораци елиминације ( $x_{21}$ )

- Жуто означене једначине су већ искоришћене

Елиминација  $x_{21}$

$$z = 5x_{11} + 5x_{12} + 3x_{13} + 6x_{21} + 4x_{22} + x_{23} \Rightarrow z = 80 - 2x_{22} - 3x_{23}$$

$$x_{11} + x_{21} = 8 \Rightarrow x_{11} = 8 - x_{21}$$

$$x_{12} + x_{22} = 5 \Rightarrow x_{12} = 5 - x_{22}$$

$$x_{13} + x_{23} = 2 \Rightarrow x_{13} = 2 - x_{23}$$

$$x_{11} + x_{12} + x_{13} = 6 \Rightarrow -x_{21} + x_{12} + x_{13} = -2 \Rightarrow -x_{21} - x_{22} - x_{23} = -9$$

$$x_{21} + x_{22} + x_{23} = 9 \Rightarrow x_{21} = 9 - x_{22} - x_{23}$$

# Решење

- После ове елиминације остаје само
$$z = 80 - 2x_{22} - 3x_{23}$$
- Ову функцију је потребно минимизовати, што се постиже узимањем максималних вредности за  $x_{22} = 5$  и  $x_{23} = 2$
- Све остале променљиве добијају се из једнакости које су искоришћене за њихово "елиминисање"
- Коначно  $z_{\min} = 64$ ,  
 $(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}) = (6,0,0,2,5,2)$
- У општем случају решавања LP проблема са једнакостима постоје две могућности
  - елиминисање једне променљиве једном једначином
  - увођење вештачке (eng: artificial) променљиве тада се може применити Dantzig алгоритам, али са обе фазе

# Основне чињенице о Dantzig simplex алгоритму

- Најчешће коришћен алгоритам у пракси за LP оптимизацију
- У зависности од полазног решења, могуће је обићи све екстреме да би се стигло до најбољег (што је еквивалентно систематском претраживању)
- Време потребно за претраживање је **експоненцијално** (у најнеповољнијим случајевима, постоји строг доказ)
- Без обзира на то, овај алгоритам добро ради у пракси
- Сложеност
  - у пракси је типично  $O(N^3)$ ,
  - теоријски експоненцијална (обилазак свих екстремних тачака, могу се конструисати примери где алгоритам мора да прође све екстреме пре него што пронађе решење)

# Имплементације Dantzig симплекс алгоритма

- MATLAB:
  - linprog,
  - intlinprog
- Mathematica:
  - LinearProgramming
- Numerical Recipes In C: The Art Of Scientific Computing: `void simplex(...)`
- Python: `scipy.optimize.linprog`

# Други алгоритми за LP

- Criss-cross algorithm  
нешто једноставнији од Dantzig simplex
- Fourier–Motzkin elimination
- Karmarkar's algorithm  
један од ретких алгоритама који је био патентиран,  
прилично компликован  
(перформансе у пракси нису боље од Dantzig симплекса)
- Методи унутрашње тачке (interior-point methods)  
проналазе решење крећући се по  
унутрашњости симплекса  
(за разлику од Dantzig алгоритма који иде по ивицама!)
- Сви алгоритми који решавају NLP проблеме могу да се  
примене (при томе, некада је потребно променити  
оптимизациону функцију  $f_0 = -\|f\|_p + \text{const.}$ )

# Задатак за вежбе

- Потребно је поставити сервере у ормаре тако да се **максимизира** рачунски капацитет система који користи све сервере у ормарима
- Сваки од 3 ормара има ограничење по броју сервера и укупној снаги сервера
- На располагању стоји различит број сервера, за сваки од 4 типа
- Одредити распоред сервера по ормарима и израчунати максимални остварени рачунски капацитет система

	Максималан број сервера	Максимална снага [W]
Ормар 1	10	6800
Ормар 2	16	8700
Ормар 3	8	4300

	Укупан број сервера на располагању	Снага појединачног сервера [W]	Рачунски капацитет појединачног сервера [GFlops]
Сервер 1	18	480	310
Сервер 2	15	650	380
Сервер 3	23	580	350
Сервер 4	12	390	285

# Поступак решавања

- Написати оптимизациону функцију и задате услове у стандардном LP облику
  - Непознате су бројеви сервера постављени у ормаре  $x_{rs}$ , где је  $r$  редни број ормара (1,2,3), а  $s$  редни број типа сервера (1,2,3,4)
- Написати програм који решава задати проблем
  - Користити Python функцију за линеарно програмирање или написати одговарајући код
  - Водити рачуна да су променљиве цели (ненегативни) бројеви! (уколико се за решење  $x_{rs}$  добије број који није цео, а различит од нуле, проверити и три мања и три већа цела броја)
- ASCII фајл уз решење би требало да садржи:
  - Оптималан распоред сервера по ормарима записан у облику  $(x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}, x_{31}, x_{32}, x_{33}, x_{34})$
  - Максималан остварени рачунски капацитет система
  - Уколико се за променљиве прво добију нецелобројна решења, записати и то решење у истом облику са одговарајућим максималним рачунским капацитетом у том случају