CAD Algorithms

Linear Programming

Mohammad Tehranipoor

ECE Department

Linear Programming (LP)

- What is LP?
  - A Linear Program (LP) is a problem that can be expressed as follows (the so-called Standard Form):

```
minimize  cx                       // Objective Function
subject to Ax = b                  // Constraints
           x >= 0
```

LP

- **x** is the vector of variables to be solved for
- **A** is a matrix of known coefficients
- **c** and **b** are vectors of known coefficients
- The expression "**cx**" is called the objective function
- The equations "**Ax=b**" are called the constraints
- The matrix **A** is generally not square, hence we can't solve an LP by just inverting **A**.

- LP tools (softwares) can handle both **minimization** and **maximization** problems.

LP

- The importance of linear programming derives in part from:
  - Its many applications
    - e.g. VLSI Physical Design
  - The existence of good general-purpose techniques (softwares) for finding optimal solutions.
    - CPLEX, LPSolve, AMPL, etc.
    - These techniques take as input only an LP in the above Standard Form, and determine a solution without reference to any information concerning the LP's origins or special structure.
    - They are fast and reliable over a substantial range of problem sizes and applications.

Integer Linear Programming (ILP)

- Integer programming (or integer linear programming) requires some or all of the variables to take **integer** (whole number) values.
- **Advantage**: Integer programs (IPs) often have the advantage of being more realistic than LPs.
  - It provides exact result
  - It is deterministic
- **Disadvantage**: IPs are much harder to solve.
- The most widely used general-purpose techniques (softwares) for solving IPs use the solutions to a series of LPs to manage the search for integer solutions and to prove optimality.

ILP vs. LP

- Variables are constructed to take integer values.
- ILP is very much harder problem than ordinary LP.
- In both theory and practice, ILP is more realistic.
- Many VLSI physical design problems are ILP problems, e.g. Floorplanning.

Tools/Softwares

- **Lpsolve**
  - http://www.geocities.com/lpsolve/
- **AMPL**
  - http://www.AMPL.com/
- **CPLEX**
  - http://www.ilog.com/products/cplex/
- There is a lot more …

Example

- Example from AMPL book:
  - Product A and B
  - Tons per hour:
    - A: 200
    - B: 140
  - Profit per ton:
    - A: $25
    - B: $30

Cont.

- Maximum tons:
  - A: 6000
  - B: 4000

- **Question**:
  - For 40 hours of production time, how many tons of A and how many tons of B should be produced to bring in the maximum profit?

Cont.

- **Answer**:
  - Variables:
    - A or X_A (tons of A?)
    - B or X_B (tons of B?)

- The total hours to produce all these tons is given by:

(hours to make a ton of A) . X_A + (hours to make a ton of B) . X_B

$(1/200) \cdot X_A + (1/140) \cdot X_B <= 40$

$0 =< X_A <= 6000$

$0 =< X_B <= 4000$

Profit: (Profit per ton of A) . X_A + (Profit per ton of B) . X_B = $25X_A + 30X_B$

Cont.

- Linear Program:

Maximize $25X_A + 30X_B$

Subject to:

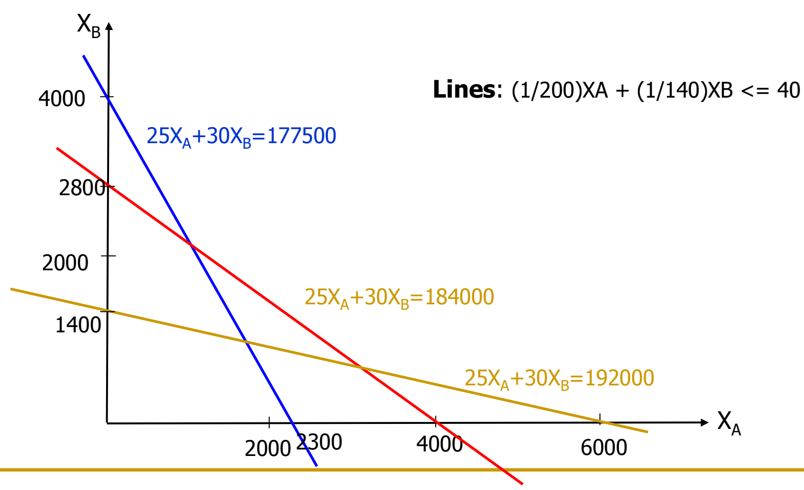
$(1/200)X_A + (1/140)X_B <= 40$

$0 =< X_A <= 6000$

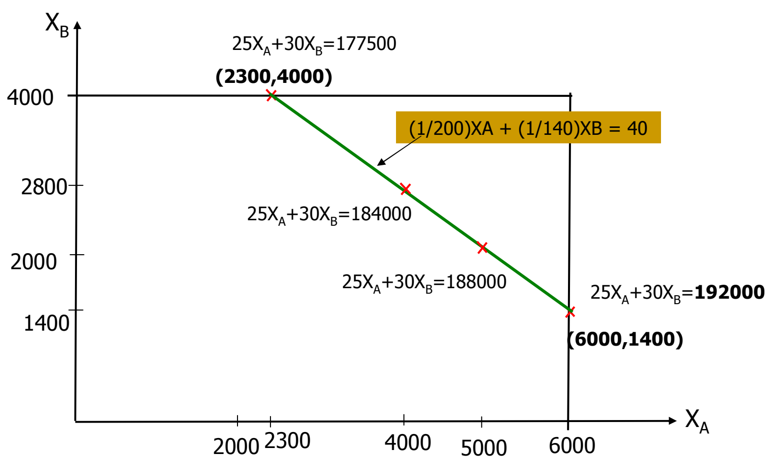
$0 =< X_B <= 4000$

Cont.

Maximize $25X_A + 30X_B$, Subject to: $(1/200)X_A + (1/140)X_B <= 40$, $0 =< X_A <= 6000$, $0 =< X_B <= 4000$

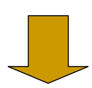


Constraint and Search Space



Linear Program in AMPL

```
Var  XA
Var  XB
Maximize Profit: 25*XA + 30*XB;
Subject to Time: (1/200)*XA + (1/140)*XB <= 40;
Subject to A Limit: 0 =< XA <= 6000;
Subject to B Limit: 0 =< XB <= 4000;
```



Store it in p1.mod

Cont.

- Then type these commands:

  ampl: model p1.mod
  ampl: solve

  2 iterations, objective: 192000

  amp1: display XA and XB

      XA = 6000
      XB = 1400

  ampl: quit

Example Problems

- Job Scheduling
  - Try the example shown in previous lecture
- Floorplanning
- Scheduling chips under test to control power
- Scheduling design with various delays
- Scan cell reordering

Non-Linear Programming (NLP)

minimize F(x)
subject to:

$g_i(x) = 0$ for $i = 1, \ldots, m1$ where $m1 >= 0$

$h_j(x) >= 0$ for $j = 1, \ldots, m$ where $m >= m1$

- Genetic and Simulated Annealing Algorithms are categorized in NLP.