Introduction to Optimization
Lecture 7

Discrete Optimization

- Discrete optimization is a generalization in which some or all the decision variables must take integer values.
- Examples:
  - Number of people, items, etc
  - YES/NO decisions (binary)

Types of Discrete Optimization

- **Integer optimization:** all the decision variables are integers and all the constraints and the objective functions are linear.
- **Binary optimization:** all the decision variables are binary and all of the constraints and the objective are linear
- **Mixed-integer optimization:** only some of the decision variables are integer or binary, and all the constraints and the objective function are linear.

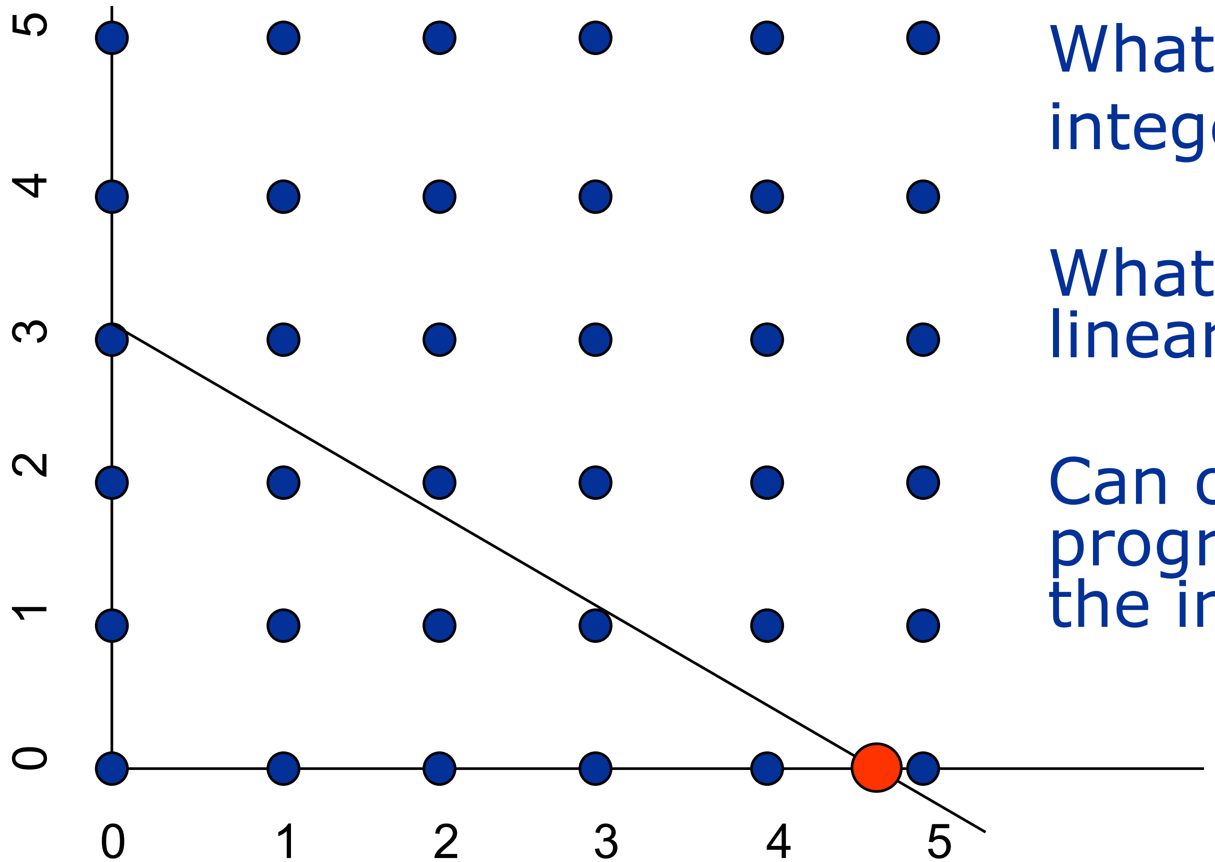
Example

- Consider the following problem

$$
\begin{aligned}
\text{maximize} \quad & 3x + 4y \\
\text{subject to} \quad & 5x + 8y \leq 24 \\
& x, y \geq 0 \text{ and integer}
\end{aligned}
$$

- What is the optimal solution?

The feasible region of an IP



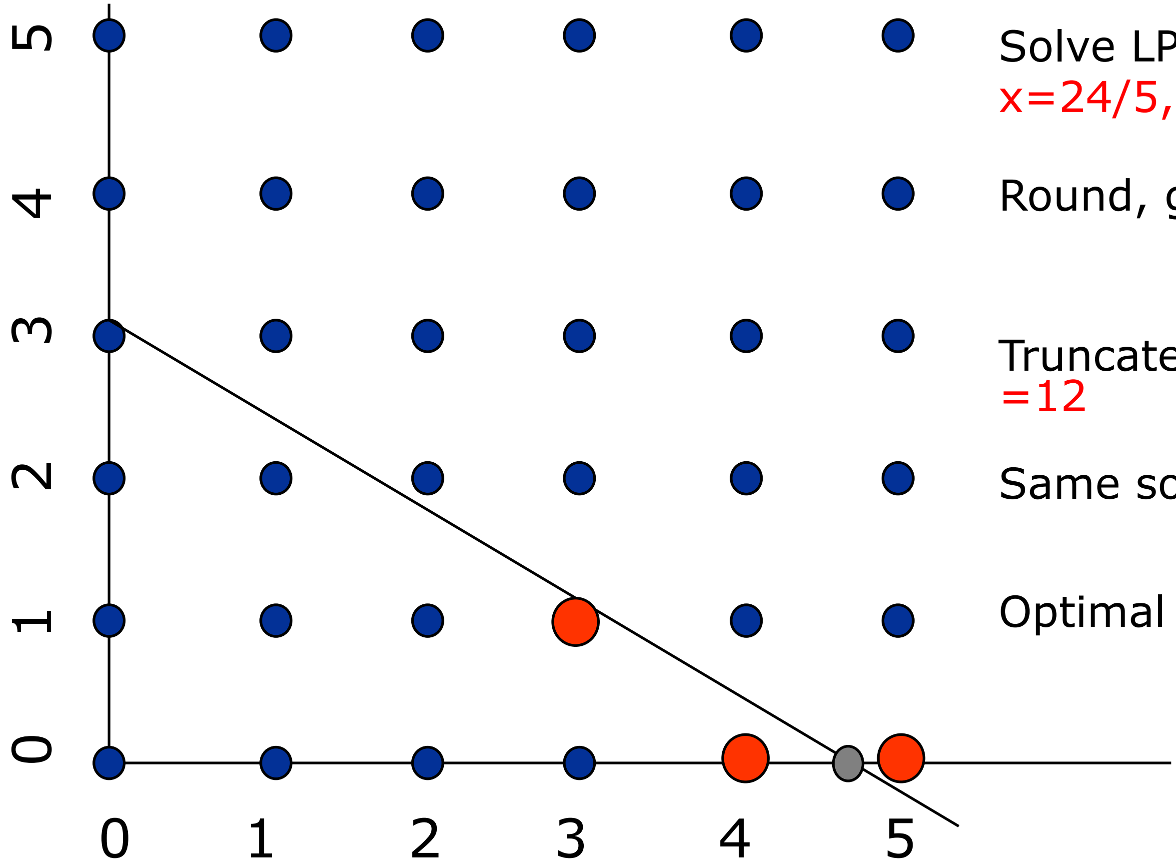
What is the optimal integer solution?

What is the optimal linear solution?

Can one use linear programming to solve the integer program?

Rounding a LP solution

A rounding technique that sometimes is useful, and sometimes not



Solve LP (ignore integrality) get $x=24/5$, $y=0$ and $Z=14\ 2/5$

Round, get $x=5$, $y=0$, infeasible!

Truncate, get $x=4$, $y=0$, and $Z=12$

Same solution value at $x=0$, $y=3$.

Optimal is $x=3$, $y=1$, and $Z^{*}=13$

Why Integer Programming

- Advantages of restricting variables to take on integer values
  - More realistic (economic indivisibilities)
  - More flexibility (use of binary variables, logical constraints)
- Disadvantages
  - More difficult to model
  - Can be <u>much</u> more difficult to solve

Binary integer variables

- Binary variables can take values of 0 or 1; typically the constraint is represented as $X_j \in \{0,1\}$
  - This is also equivalent to $0 \leq X_j \leq 1$ *and integer*
- Binary variables are very useful in modeling several business situations
  - Logical constraints (e.g., if-then-else, go-no/go decisions)
  - Application areas include supply-chain optimization models (transportation, facility location), financial models (budget models), and many more

Project Selection Problem

- Stockco has $14,000,000 available for investment and is considering five projects. Find the solution which maximizes the total return subject to the budget constraint

Project	Return $ million	Unit Price $ million
1	6	5
2	9	7
3	5	4
4	4	3
5	7	4

Project Selection Problem

Project	Return $ million	Unit Price $ million
1	6	5
2	9	7
3	5	4
4	4	3
5	7	4

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{rl} \max & 6x_1 + 9x_2 + 5x_3 + 4x_4 + 7x_5 \\ \text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 \leq 14 \\ & x_1, x_2, x_3, x_4, x_5 \text{ binary} \end{array}$$

Project Selection Problem

- A typical projection problem (or Knapsack Problem)

n: projects, total budget b

a_j: returns of projects j

c_j: cost of projects j

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \max \quad & a_1 x_1 + \ldots + a_n x_n \\ \text{s.t.} \quad & c_1 x_1 + \ldots + c_n x_n \leq b \\ & x_j \text{ binary} \end{aligned}$$

Project Selection Problem

Project	Decision Variable	Optimal Value
1	6	0
2	9	1
3	5	0
4	4	1
5	7	1

Earnings:

$$9 + 4 + 7 = \$20\text{millions}$$

Common Relations

- At most one event can occur

$$x_1 + x_2 + x_3 \leq 1$$
$$x_1, x_2, x_3 \text{ binary}$$

- Example: Suppose we can only choose project 1 or project 3 but not both.

$$\begin{array}{rl} \max & 6x_1 + 9x_2 + 5x_3 + 4x_4 + 7x_5 \\ \text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 \leq 14 \\ & x_1 + x_3 \leq 1 \\ & x_1, x_2, x_3, x_4, x_5 \text{ binary} \end{array}$$

Common Relations

- Neither event occurs or both events occur.

$$\begin{array}{l} x_2 = x_1 \\ x_1, x_2 \text{ binary} \end{array}$$

- Example: Project 4 and 5 must be invested together or left out together.

$$\begin{array}{rl} \max & 6x_1 + 9x_2 + 5x_3 + 4x_4 + 7x_5 \\ \text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 \leq 14 \\ & x_4 = x_5 \\ & x_1, x_2, x_3, x_4, x_5 \text{ binary} \end{array}$$

Common Relations

- If event 2 occurs, then event 1 must occur. Note that event 1 can occur without event 2.

$$x_2 \leq x_1$$
$$x_1, x_2 \text{ binary}$$

- Example: If Project 4 is chosen, then Project 5 must also be chosen.

$$\begin{aligned} \text{max} \quad & 6x_1 + 9x_2 + 5x_3 + 4x_4 + 7x_5 \\ \text{s.t.} \quad & 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 \leq 14 \\ & x_4 \leq x_5 \\ & x_1, x_2, x_3, x_4, x_5 \text{ binary} \end{aligned}$$

Example: NUS Course Scheduling

- You would like to determine your optimal course schedule for your first two terms at NUS
- You have created a list of the 20 courses that most interest you, with interest levels from 3 to 5, and you want to maximize your total interest level

Example: NUS Course Scheduling

- Constraints:
  - You can only take a class if you have completed or are currently taking all the courses that are pre-requisite for a particular course
  - In the Fall Semester, you must take at least 3 of the following courses: ADM, Micro-Economics, Finance Theory, Accounting, Business Communication
  - If you take Financial Engineering you can't take Options and Futures because the two courses overlap
  - You want to take at least one course in Marketing and at least one in Operations Management
  - You can't take the same class twice

Example: NUS Course Scheduling

- The data

Course Number	Course Title	Semester Offered	Course Prerequistes	Interest Level
1	**ADM**	Fall		**5**
2	**Micro-Economics**	Fall		**5**
3	**Finance Theory**	Fall and Spring		**4**
4	**Strategy I**	Fall		**4**
5	**Strategy II**	Spring	4	**4**
6	**Accounting I**	Fall		**3**
7	**Accounting II**	Spring	2, 6	**3**
8	**Finacial Engineering**	Spring	1, 3	**5**
9	**Statistics**	Spring	1	**4**
10	**Operations Management I**	Fall		**4**
11	**Operations Management II**	Spring	1, 10	**4**
12	**Marketing I**	Fall		**3**
13	**Marketing II**	Spring	9, 12	**3**
14	**Options and Futures**	Spring	3	**5**
15	**Information Technology I**	Fall		**4**
16	**Information Technology II**	Spring	15	**4**
17	**Entrepreneurship**	Spring	4	**4**
18	**New Product Development**	Spring	10,12,17	**3**
19	**Organizational Processes**	Fall	4	**3**
20	**Business Communications**	Fall		**5**

Example: NUS Course Scheduling

- Let $F_1, \ldots, F_{20}$ be binary variables indicating which of the 20 courses you decide to take during the fall term

  e.g. $F_1 = 1$ means that you take course number 1 in the fall

- Similarly, let $S_1, \ldots, S_{20}$ be binary variables indicating whether or not you take any of the 20 courses during the spring term

- Let $OF_1, \ldots, OF_{20}$ and $OS_1, .., OS_{20}$ be binary constants indicating whether each of the classes is offered in the fall and spring terms, respectively

  e.g. $OF_1 = 1$, $OS_1 = 0$

- Finally, let $I_1, .., I_{20}$ be the interest level you assigned to each of the 20 courses

(**GREEN** will be data and **RED** will be variables)

Example: NUS Course Scheduling

Objective and Constraints

- Objective: maximize total interest

$$\text{Max } I_1*(F_1+S_1)+\ldots+I_{20}*(F_{20}+S_{20})$$

Constraints:

- Cannot take a course twice: $F_1 + S_1 \le 1, \ldots, F_{20} + S_{20} \le 1$
- Fall course offering: $F_1 \le OF_1, \ldots, F_{20} \le OF_{20}$
- Spring course offering: $S_1 \le OS_1, \ldots, S_{20} \le OS_{20}$
- Fall term maximum: $F_1 + \ldots + F_{20} \le 5$
- Spring term maximum: $S_1 + \ldots + S_{20} \le 5$

Example: NUS Course Scheduling

- Fall term requirement: $F_1 + F_2 + F_3 + F_6 + F_{20} \geq 3$
- Financial or Options: $S_{14} + S_8 \leq 1$
- Marketing: $F_{12} + S_{13} \geq 1$
- O.M.: $F_{10} + S_{11} \geq 1$
- Binary: $F_1, \ldots, F_{20}, S_1, \ldots, S_{20} = 0 \text{ or } 1$

More constraints

- Pre-requisites
- Prerequisites course 5: $S_5 \leq F_4$
- Prerequisites course 7: $S_7 \leq F_2$, $S_7 \leq F_6$
- Prerequisites course 8: $S_8 \leq F_1$, $S_8 \leq F_3 + S_3$
- Prerequisites course 9: $S_9 \leq F_1$
- Prerequisites course 11: $S_{11} \leq F_1$, $S_{11} \leq F_{10}$
- Prerequisites course 13: $S_{13} \leq S_9$, $S_{13} \leq F_{12}$
- Prerequisites course 14: $S_{14} \leq F_3 + S_3$
- Prerequisites course 16: $S_{16} \leq F_{15}$
- Prerequisites course 17: $S_{17} \leq F_4$
- Prerequisites course 18: $S_{18} \leq F_{10}$, $S_{18} \leq F_{12}$, $S_{18} \leq S_{17}$
- Prerequisites course 19: $F_{19} \leq F_4$

More Common Relations

- Limit range of variable, y. If event does not occur, then y=0. Otherwise, $0 \leq y \leq M$ if event occurs.

$$0 \leq y \leq Mx$$
$$x \text{ binary}$$

- M is a large number that variable y can never attain
- Usage Examples:
  - Stopping production flow if one decides to close down a production plant.

Fixed cost problem

- Cost of producing a product follows the following function

$$f(x) = \begin{cases} 10 + 5x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Fixed cost problem

- A company is considering making 3 different products. Let x_1, x_2, x_3 denote the quantity of each product to be produced.

Product	Fixed Cost	Unit Profit
1	\$ 100 if $x_1 > 0$ \$ 0 if $x_1 = 0$	\$ 5
2	\$ 150 if $x_2 > 0$ \$ 0 if $x_2 = 0$	\$ 7
3	\$ 75 if $x_3 > 0$ \$ 0 if $x_3 = 0$	\$ 4

Fixed cost problem

- The production constraints are :

$$4x_1 + 6x_2 + x_3 \leq 2000$$
$$2x_1 + 2x_2 + 3x_3 \leq 1500$$

- Formulate a model for determining the maximum profit production policy.

Fixed cost problem

- How about this?

$$
\begin{aligned}
\max \quad & 5x_1 - 100y_1 + 7x_2 - 150y_2 + 4x_3 - 75y_3 \\
\text{s.t.} \quad & 4x_1 + 6x_2 + x_3 \leq 2000 \\
& 2x_1 + 2x_2 + 3x_3 \leq 1500 \\
& x_1 \leq My_1 \\
& x_2 \leq My_2 \\
& x_3 \leq My_3 \\
& x_1, x_2, x_3 \geq 0 \\
& y_1, y_2, y_3 \text{ binaries}
\end{aligned}
$$

- How should be choose M?

Sudoku

- Developed by Howard Garns and first published in 1979
- Popularized in Japan since 1980’s
- Rules: to fill a 9x9 grid with digits so that
  - Each column contains all the digits from 1 to 9
  - Each row contains all the digits from 1 to 9
  - Each 3x3 block contains all the digits from 1 to 9

- Many algorithms available
  - One by PM Lee

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku

Lee Hsien Loong
Public Figure · 869,723 Likes · May 4 · Edited ·

Liked

I told the Founders Forum two weeks ago that the last computer program I wrote was a Sudoku solver, written in C++ several years ago (http://bit.ly/1DMK5Zk). Someone asked me for it. Here is the source code, the exe file, and a sample printout - http://bit.ly/1zAXbua

The program is pretty basic: it runs at the command prompt, in a DOS window. Type in the data line by line (e.g. 1-3-8---6), then the solver will print out the solution (or all the solutions if there are several), the number of steps the program took searching for the solution, plus some search statistics.

For techies: the program does a backtrack search, choosing the next cell to guess which minimises the fanout.

Here's a question for those reading the source code: if x is an (binary) integer, what does (x & -x) compute?

Hope you have fun playing with this. Please tell me if you find any bugs! – LHL

#SmartNation

===========

Answer: As several of you noted, (x & –x) returns the least significant '1' bit of x, i.e. the highest power of two that divides x. This assumes two's complement notation for negative numbers, as some of you also pointed out. e.g. if x=12 (binary 1100), then (x & -x) = 4 (binary 100). I didn't invent this; it is an old programming trick. 🙂

===========

Update: A few people suggested that I add a licence to the code. Have added it in the Google Drive folder.

C:\Users\pmolh\Documents\CPP\Sudoku2.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

Sudoku2.cpp

```cpp

void Place(int S)
{
    LevelCount[S]++;
    Count++;

    if (S >= 81) {
       Succeed();
       return;
    }

    int S2 = NextSeq(S);
    SwapSeqEntries(S, S2);

    int Square = Sequence[S];

    int     BlockIndex = InBlock[Square],
            RowIndex = InRow[Square],
            ColIndex = InCol[Square];

    int     Possibles = Block[BlockIndex] & Row[RowIndex] & Col[ColIndex];
    while (Possibles) {
          int valbit = Possibles & (-Possibles); // Lowest 1 bit in Possibles
          Possibles &= ~valbit;
          Entry[Square] = valbit;
          Block[BlockIndex] &= ~valbit;
          Row[RowIndex] &= ~valbit;
          Col[ColIndex] &= ~valbit;

          Place(S + 1);

          Entry[Square] = BLANK; // Could be moved out of the Loop
          Block[BlockIndex] |= valbit;
          Row[RowIndex] |= valbit;
          Col[ColIndex] |= valbit;
    }

    SwapSeqEntries(S, S2);
}


int main(int argc, char* argv[])
{
    int i, j, Square;

    for (i = 0; i < 9; i++)
        for (j = 0; j < 9; j++) {
            Square = 9 * i + j;
            InRow[Square] = i;
            InCol[Square] = j;
            InBlock[Square] = (i / 3) * 3 + ( j / 3);
        }


    for (Square = 0; Square < 81; Square++) {
        Sequence[Square] = Square;
        Entry[Square] = BLANK;
        LevelCount[Square] = 0;
    }

    for (i = 0; i < 9; i++)
        Block[i] = Row[i] = Col[i] = ONES;

    ConsoleInput();
    Place(SeqPtr);
    printf("\n\nTotal Count = %d\n", Count);

    return 0;
}

```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 1 Col: 1 Sel: 0 Lines: 223 Length: 4629 Insert Done parsing in 0 second

Sudoku – IP Formulation

- Constraints:
  - Only one *k* in each column
  - Only one *k* in each row
  - Only one *k* in each 3x3 block
  - All cells must be filled
  - The set of initially given numbers cannot be changed
  - Binary

Sudoku – IP Formulation

- Objective ???
  - Nothing to maximize or minimize
  - Just looking for a feasible solution => **Feasibility Problems**

- Decision variables: x_{ijk} --- whether cell (i, j) contains digit k (binary)

Applications: Sudoku – IP Formulation

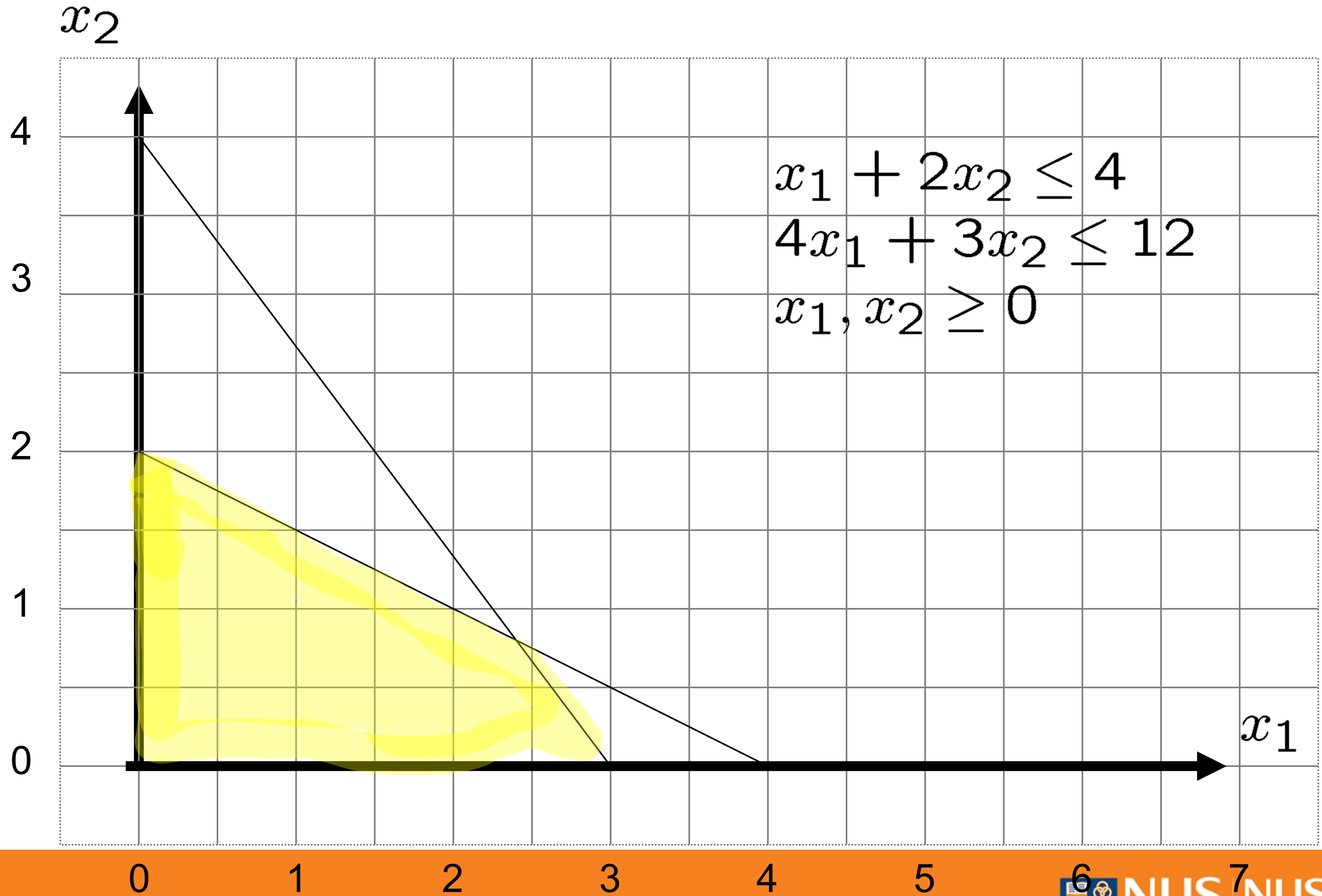
$$
\begin{array}{llll}
\min & \mathbf{0}'\boldsymbol{x} & & \\
\text{s.t.} & \displaystyle\sum_{i=1}^{9} x_{ijk} = 1, & \forall j,k \in \{1,\ldots,9\} & : \text{ one k in each column} \\
& \displaystyle\sum_{j=1}^{9} x_{ijk} = 1, & \forall i,k \in \{1,\ldots,9\} & : \text{ one k in each row} \\
& \displaystyle\sum_{j=3(q-1)+1}^{3q} \ \sum_{i=3(p-1)+1}^{3p} x_{ijk} = 1, & \forall k \in \{1,\ldots,9\}, p,q \in \{1,2,3\} & : \text{ one k in each submatrix} \\
& \displaystyle\sum_{k=1}^{9} x_{ijk} = 1, & \forall i,j \in \{1,\ldots,9\} & : \text{ position must be filled} \\
& x_{ijk} = 1 \ \ \forall (i,j,k) \in G & & : \text{ elements set to on} \\
& x_{ijk} \in \{0,1\} & \forall i,j,k \in \{1,\ldots,9\} &
\end{array}
$$

Solution Approach in Discrete Optimization

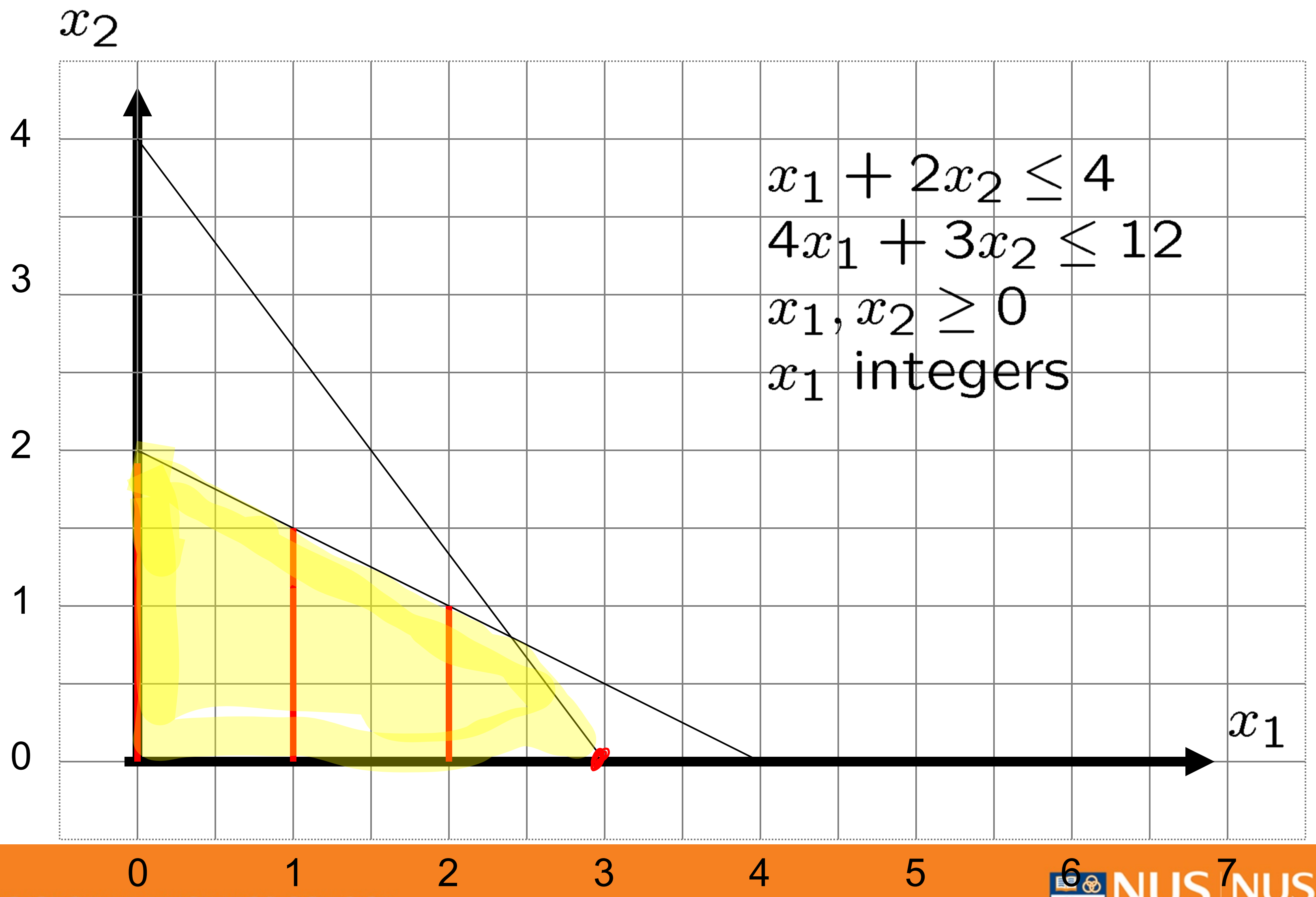
- Mixed Integer Programming

$$
\begin{array}{rll}
\text{min(max)} & c_1x_1 + \ldots + c_nx_n & \\
\text{s.t.} & a_{11}x_1 + \ldots + a_{1n}x_n = b_1 & \text{Equality constraints} \\
 & \vdots & \\
 & a_{41}x_1 + \ldots + a_{4n}x_n \leq b_4 & \text{Inequality constraints} \\
 & \vdots & \\
 & a_{81}x_1 + \ldots + a_{8n}x_n \geq b_8 & \text{Inequality constraints} \\
 & \vdots & \\
 & x_1, \ldots, x_n \geq 0, & \text{Nonnegative constriants} \\
 & x_1, .., x_k \text{ integer} & \text{Integer constriants}
\end{array}
$$

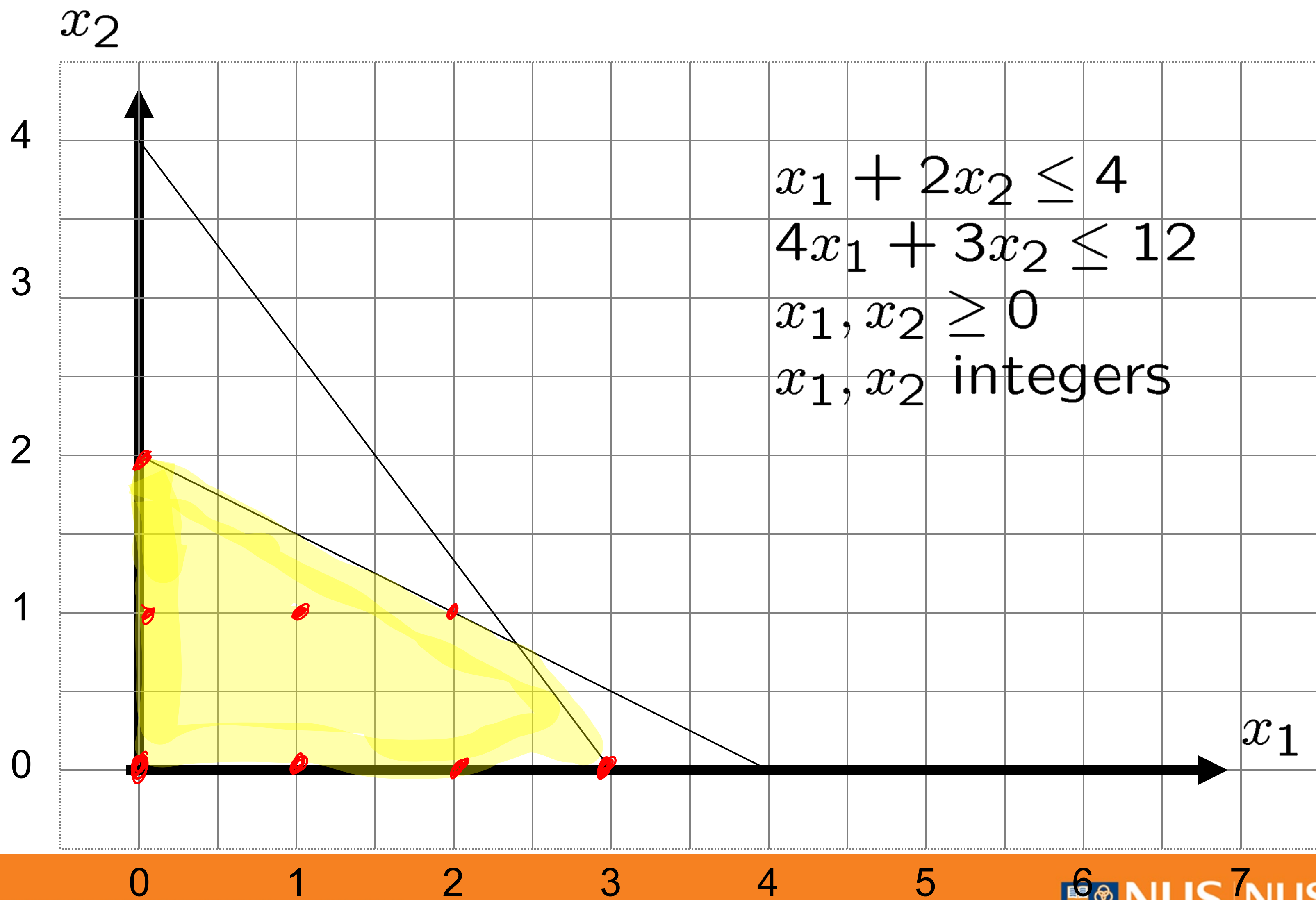
Geometric Interpretation



Geometric Interpretation

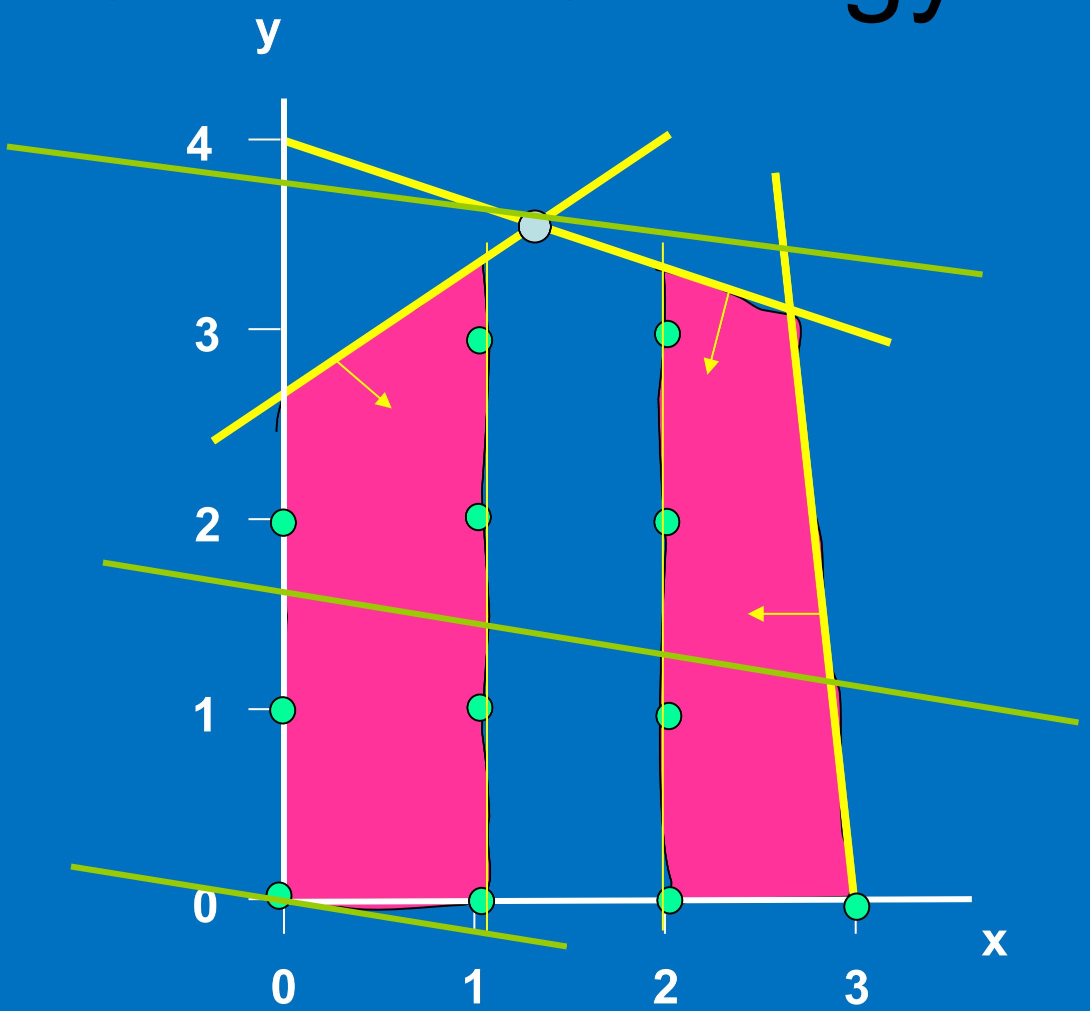


Geometric Interpretation



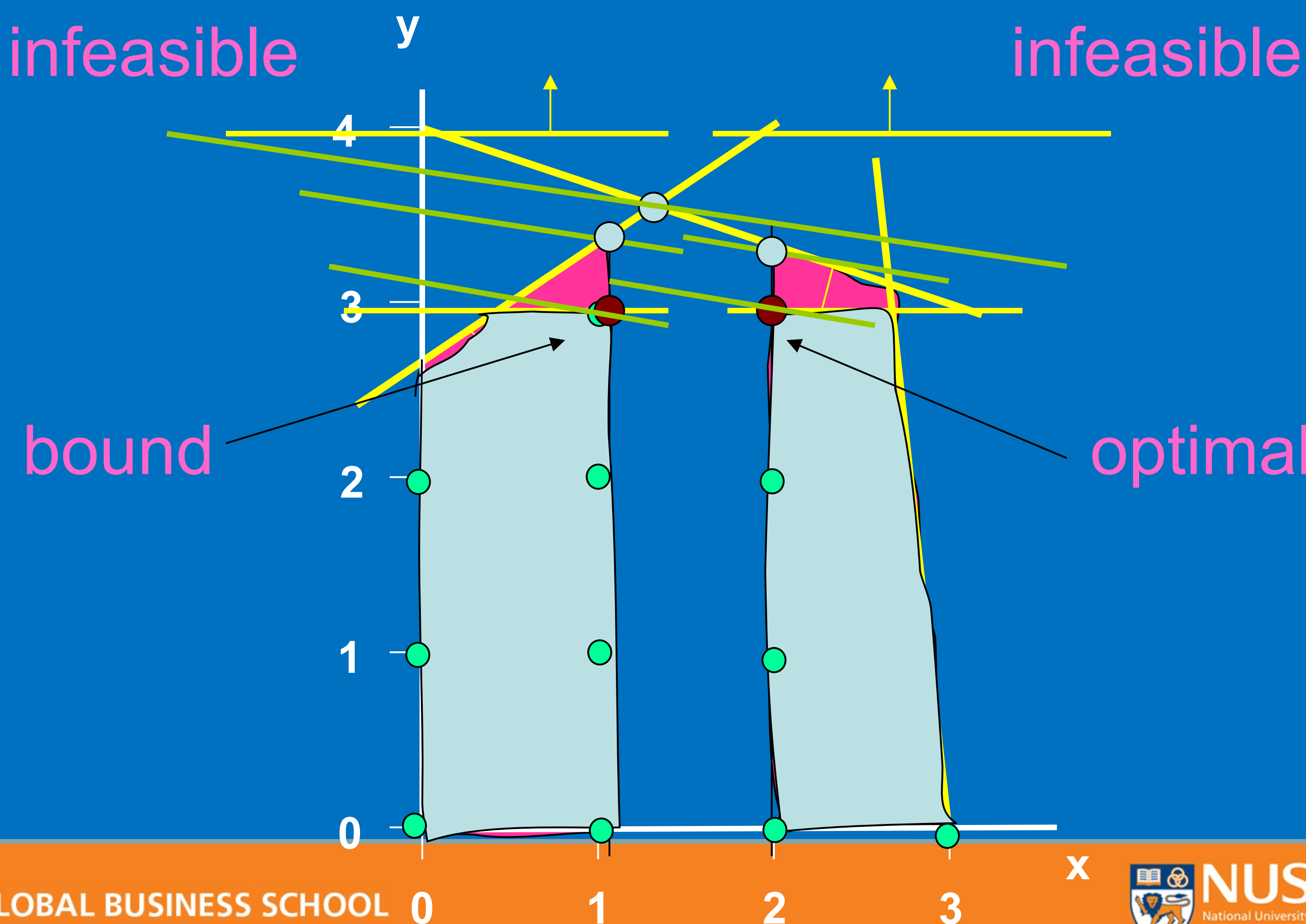
$$x_1 + 2x_2 \leq 4$$
$$4x_1 + 3x_2 \leq 12$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \text{ integers}$$

Solution Strategy



Solving the linear relaxation problem provides a bound. Also...

Branch and Bound



Computational Issues in Discrete Optimization

– Bad news: Integer programming are generally hard to solve.
– How hard is hard? Generally very hard!!!
  • Clay Millennium Challenge
  • http://www.claymath.org/millennium/P_vs_NP/

Computational Issues in Discrete Optimization

- Good news:
  – State of the art solvers such as CPLEX are able to solve many reasonable sized problem of practical importance.
- Why do we need to discrete optimization?
  – Very powerful modeling methods!!
  – CPLEX report that more than 90% of their clients’ problems are MIP
  – Sudoku