23 June

## Integer programming [P&S Chapter 13]

Defn. An integer linear program (ILP) is an optimization problem that has the form of a linear program with the additional restriction that the values of all variables must be integers.

- Note that this means an ILP is not an LP, because an LP cannot have such restrictions.

Defn. A mixed integer linear program (MILP) is an optimization problem that has the form of a linear program with the additional restriction that the values of some of the variables must be integers.

- So a MILP can have some variables that are constrained to be integers and other variables that are not.

Notes.

- Often "linear" is left implicit, so these problems are called IP and MIP.
- The distinction between the two is not really important, so often "IP" is used to include them both, or "MIP" to include them both.

## Why integer-valued variables?

- Perhaps the variables represent quantities of something that cannot be subdivided — e.g., numbers of airplanes to be assigned to routes.

- More often, however, integer-valued variables are used intentionally in order to model yes-no decisions, combinatorial constraints, nonlinearities, etc. These are <u>not</u> just quantities of indivisible things.

## Why not just round to nearest integer?

<u>Example.</u>

$$\begin{aligned} \max\ & 3x_1 + 13x_2 \\ \text{s.t. } & 2x_1 + 9x_2 \le 40 \\ & 11x_1 - 8x_2 \le 82 \end{aligned}$$

$$\text{Domains.} \begin{cases} x_1 \ge 0, \quad x_2 \ge 0 \\ x_1, x_2 \text{ integer.} \end{cases}$$

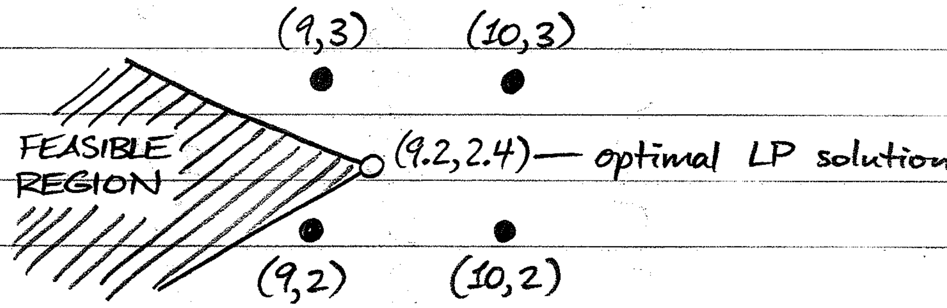
Remove the integer domain restrictions to get the <u>LP relaxation</u>:

$$\begin{aligned} \max\ & 3x_1 + 13x_2 \\ \text{s.t. } & 2x_1 + 9x_2 \le 40 \\ & 11x_1 - 8x_2 \le 82 \\ & x_1 \ge 0, \quad x_2 \ge 0. \end{aligned}$$

23 June

Optimal solution to LP relaxation is
$x_1 = 9.2$, $x_2 = 2.4$.

But <u>none</u> of the four nearest integer points (9,2), (9,3), (10,2), or (10,3) is feasible!




The optimal <u>integer</u> solution turns out to be $x_1 = 2$, $x_2 = 4$, which is nowhere near the optimal LP solution and certainly cannot just be obtained by rounding.

— Anyway, we should not expect rounding to work, because LPs are easy to solve, and rounding is easy, but using integer variables to model yes-no decisions or combinatorial constraints allows us to formulate many <u>hard</u> problems as IPs.

## Example. Knapsack problem.

Indiana Jones problem from first problem set:

- Knapsack has capacity 20.
- Available items:

| Item | Weight | Value ($1000's) |
|---|---|---|
| A | 4 | 4 |
| B | 6 | 9 |
| C | 7 | 7 |
| D | 13 | 17 |
| E | 5 | 5 |
| F | 2 | 2 |
| G | 10 | 15 |
| H | 5 | 8 |
| I | 3 | 4 |
| J | 8 | 11 |

- Maximize value without exceeding capacity of knapsack.

Variables $x_A, x_B, \ldots, x_J \in \{0,1\}$: $x_i = 1$ means item $i$ should be taken, $x_i = 0$ means not.

IP: $\max\ 4x_A + 9x_B + 7x_C + 17x_D + 5x_E + 2x_F + 15x_G + 8x_H + 4x_I + 11x_J$

$\text{s.t. } 4x_A + 6x_B + 7x_C + 13x_D + 5x_E + 2x_F + 10x_G + 5x_H + 3x_I + 8x_J \leq 20$

$x_i \in \{0,1\}$ for all $i$.

23 June

<u>Note.</u> We have some new variable domains in integer programming.

$$\left.\begin{array}{l} x \geq 0 \\ x \leq 0 \\ x \text{ unrestricted} \end{array}\right\} \text{Good old LP domains}$$

$$\left.\begin{array}{l} x \geq 0,\ x \text{ integer} \\ x \leq 0,\ x \text{ integer} \\ x \text{ unrestricted},\ x \text{ integer} \end{array}\right\} \text{New IP domains}$$

$x \in \{0,1\}$ ⟵ Extremely common special case: this is a <u>Boolean</u> (or <u>binary</u>) variable, often representing a yes-no decision.

## <u>Solving IPs in Maple</u>

— If <u>all</u> variables must be nonnegative integers, use assume=nonnegint instead of assume=nonnegative as an option to LPSolve.

In general:

— The <u>assume</u> option can be set to nonnegative, integer, binary, or nonnegint. This will apply to all variables that do not have domains explicitly set elsewhere. By default, none of these properties are assumed.

- To explicitly specify that some variables must be integers, use the option <u>integervariables</u>, like integervariables = [x1, x3, x4].

- To explicitly specify that some variables must be binary (i.e., values in $\{0, 1\}$), use the option <u>binaryvariables</u> (like integervariables).

- Occasionally you may get an error message saying that "maximal depth of branch and bound search is too small." (We will see what this means in the next few days.)
<u>If</u> this happens, you can increase the depth limit with the <u>depthlimit</u> option (e.g., depthlimit = 10) and try again.

23 June

## More IP formulation.

Guideline for choosing variables:

- Variables represent decisions to be made.
- In LP, those decisions were generally quantities: how many acres of wheat, how many overtime hours, how much flow along an arc, etc.
- Now in IP we also have yes-no decisions. These are not quantities. Such decisions can be modeled with $\{0,1\}$ variables.
- Many IP formulations consist entirely of these two kinds of variables:
  - Quantities are represented by non-integer variables;
  - Yes-no decisions are represented by $\{0,1\}$ variables.
- So try to decompose the problem into a collection of quantity decisions and yes-no decisions.

⚠ IMPORTANT: Remember that everything still has to be linear! So you can't multiply variables together.

## Example. Set covering.

- Six communities: A, B, C, D, E, F.
- Must build fire stations in one or more communities so that all communities are served by a fire station no more than 15 minutes away.
- Travel times between communities:

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | – | 5 | 25 | 10 | 15 | 20 |
| B | | – | 20 | 5 | 20 | 10 |
| C | (symmetric) | | – | 25 | 10 | 15 |
| D | | | | – | 30 | 20 |
| E | | | | | – | 10 |
| F | | | | | | – |

- Costs of building in each community (in \$1000's):

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 100 | 130 | 90 | 140 | 95 | 110 |

- Minimize total building cost.

Variables: $x_A, x_B, x_C, x_D, x_E, x_F \in \{0,1\}$: whether to build in each community.

$$
\begin{array}{llr}
\min & 100x_A + 130x_B + 90x_C + 140x_D + 95x_E + 110x_F & \\
\text{s.t.} & x_A + x_B + x_D + x_E \geq 1 & \text{[must serve A]} \\
& x_A + x_B + x_D + x_F \geq 1 & \text{[must serve B]} \\
& x_C + x_E + x_F \geq 1 & \text{[must serve C]} \\
& x_A + x_B + x_D \geq 1 & \text{[must serve D]} \\
& x_A + x_C + x_E + x_F \geq 1 & \text{[must serve E]} \\
& x_B + x_C + x_E + x_F \geq 1 & \text{[must serve F]}
\end{array}
$$

All variables $\{0,1\}$.

IP — 5

23 June

Example. Start-up costs.

— A piece of equipment must be brought online (started up, repaired, purchased, etc.) before it is used. This has some cost $S$.

— If the equipment is brought online, it can be used up to some capacity $M$ (e.g., it can produce at most $M$ units).

Variables:

- $b \in \{0,1\}$: whether to bring equipment online.
- $x \geq 0$: quantity processed by equipment.

Expression ~~for~~ cost: $Sb$

Capacity constraint: $x \leq Mb$