Lecture 03 – Integer Programming

Optimisation CT5141

James McDermott

University of Galway

Overview

Integer programming

In **integer programming** (IP) everything is exactly the same as in linear programming, except there is a new type of constraint: **decision variables are integer**.

We drop the **divisibility** assumption of LP.

There are many important applications (and different algorithms).

Integer programming

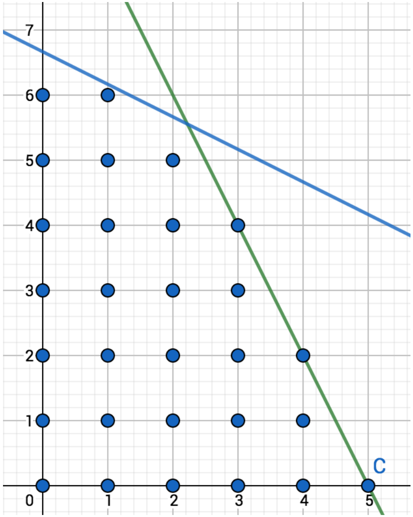
In **integer programming** (IP) everything is exactly the same as in linear programming, except there is a new type of constraint: **decision variables are integer**.

We drop the **divisibility** assumption of LP.

There are many important applications (and different algorithms).

Sometimes it is called **integer linear programming** (ILP) because the objective and constraints still have to be linear.

IP: feasible area



In integer programming, the feasible area is a grid of points inside a polygon.

Machine press example

- Machine shop (manufacturer) will buy new equipment of two types: presses and lathes.
- Can only buy an integer number of each.
- Marginal profitability: each press €100/day; each lathe €150/day.
- Resource constraints: budget €40,000, 200m^2 floor space.
- Each press requires 15m^2 and costs €8000
- Each lathe requires 30m^2 and costs €4000

Machine press example

DVs: let x_1 = number of presses, x_2 = number of lathes, both integer.

Maximise:

$$100x_1 + 150x_2$$

Subject to:

$$\begin{aligned} 8000x_1 + 4000x_2 &\leq 40000 \\ 15x_1 + 30x_2 &\leq 200\text{m}^2 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 & \quad \text{integer} \end{aligned}$$

Integer constraints

- It is only the decision variables x_1 and x_2 which are constrained to be integer
- Don't confuse this with the **data** of the problem
- E.g. the objective function coefficients 100 and 150 happen to be integers here but could be real in another problem.

Maximise:

$$100x_1 + 150x_2$$

Subject to:

$$\begin{aligned} 8000x_1 + 4000x_2 &\leq 40000 \\ 15x_1 + 30x_2 &\leq 200\text{m}^2 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\quad \text{integer} \end{aligned}$$

Why don’t we just…

Why don’t we just ignore the integer issue, and solve the problem using LP, and then round our solution off to the nearest integer value of each decision variable?

This is called the **LP relaxation** because we **relax** the integer constraint.

- Sometimes, especially when decision variables will have large values, we **can** do this, and get a good solution

Why don't we just…

Why don't we just ignore the integer issue, and solve the problem using LP, and then round our solution off to the nearest integer value of each decision variable?

This is called the **LP relaxation** because we **relax** the integer constraint.

- Sometimes, especially when decision variables will have large values, we **can** do this, and get a good solution
- (But we must round off carefully to stay inside constraints)

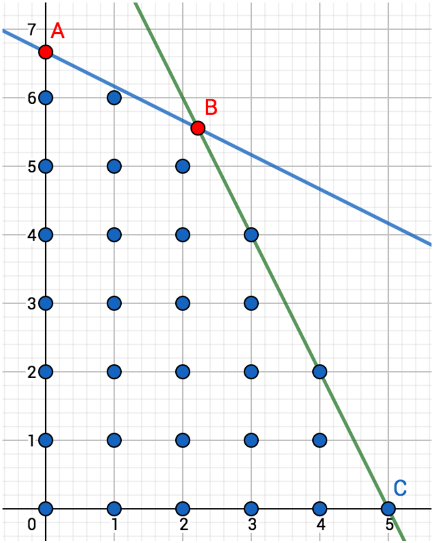
Why don't we just…

Why don't we just ignore the integer issue, and solve the problem using LP, and then round our solution off to the nearest integer value of each decision variable?

This is called the **LP relaxation** because we **relax** the integer constraint.

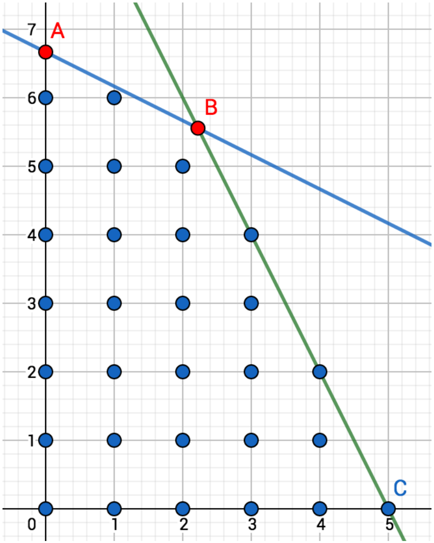
- Sometimes, especially when decision variables will have large values, we **can** do this, and get a good solution
- (But we must round off carefully to stay inside constraints)
- Sometimes, especially when decision variables will have small values, rounding off will give a **sub-optimal solution**!

Machine press example: LP relaxation



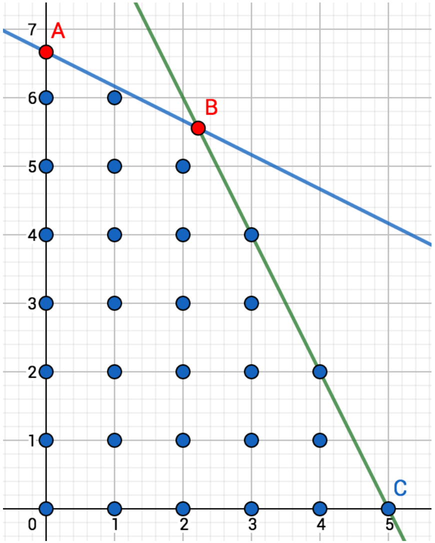
- **Let's ignore the integer constraint for now**

Machine press example: LP relaxation



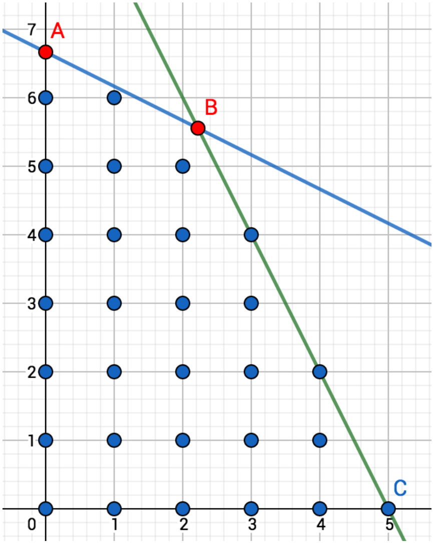
- **Let's ignore the integer constraint for now**
- This is called **solving the LP relaxation**

Machine press example: LP relaxation



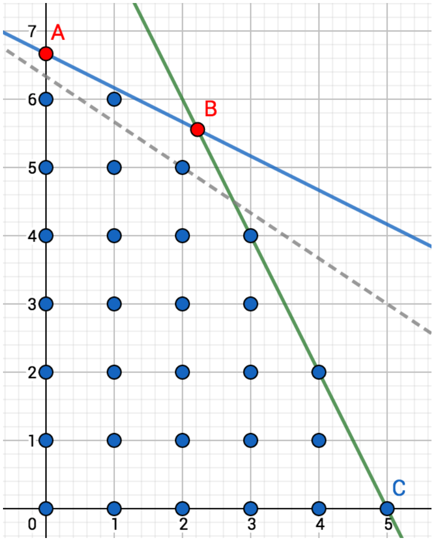
- **Let's ignore the integer constraint for now**
- This is called **solving the LP relaxation**
- This gives the optimum $B = (2.22, 5.56)$, with $f(B) = 1,055.56$.

Machine press example: LP relaxation



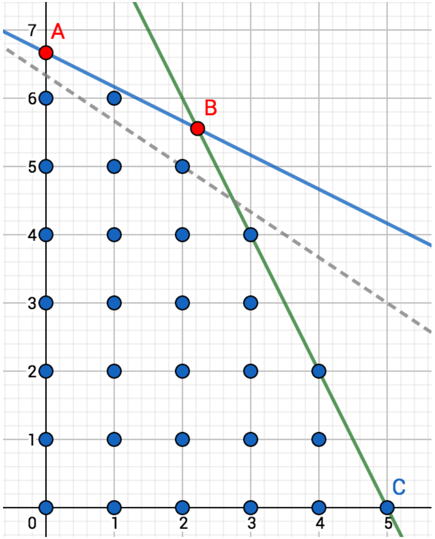
- **Let's ignore the integer constraint for now**
- This is called **solving the LP relaxation**
- This gives the optimum $B = (2.22, 5.56)$, with $f(B) = 1,055.56$.
- **Rounding B off** then gives $(2, 5)$ with $f(2, 5) = 950$ (obeying integer constraint)

Machine press example: LP relaxation



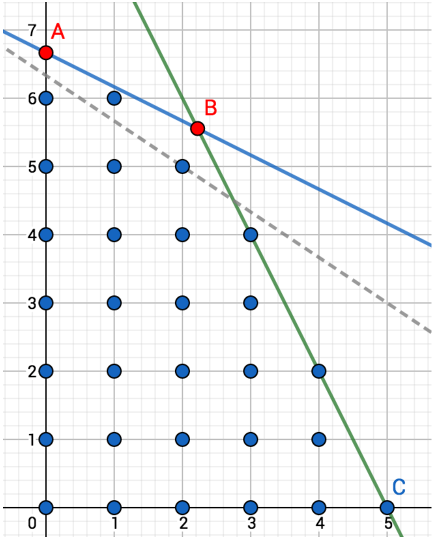
- We add in the LOEP@950 for illustration

Machine press example: LP relaxation



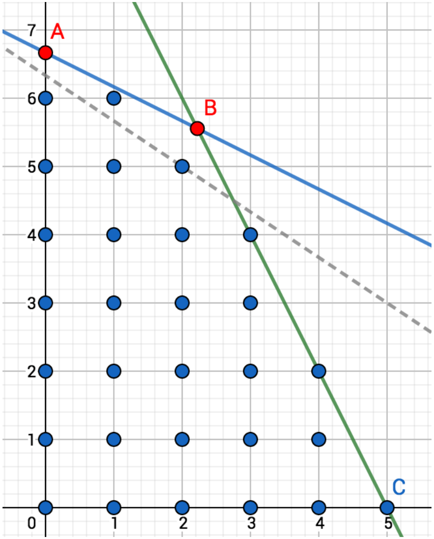
- We add in the LOEP@950 for illustration
- But $(1, 6)$ is better! $f(1, 6) = 1000$. We could not achieve this by rounding.

Machine press example: LP relaxation



- We add in the LOEP@950 for illustration
- But (1, 6) is better! $f(1, 6) = 1000$. We could not achieve this by rounding.
- Thus, LP relaxation gives us an **upper bound** on optimum profits, but not necessarily the true solution

Machine press example: LP relaxation



- We add in the LOEP@950 for illustration
- But $(1, 6)$ is better! $f(1, 6) = 1000$. We could not achieve this by rounding.
- Thus, LP relaxation gives us an **upper bound** on optimum profits, but not necessarily the true solution
- (For a **minimisation** problem, the LP relaxation gives a **lower** bound.)

LP Relaxation

Solving LP Relaxation and rounding off doesn't find the solution to an IP problem. But it is used as a **component** of the IP **branch and bound** algorithm we will see next week.

Overview

Binary decision variables

Binary decision variables are common in IP – sometimes called BIP.

LP relaxation and rounding fails completely!

“How many” versus “whether”

In LP and IP, a decision variable x_i often means **how many** of some quantity, e.g. how many of product i should we manufacture?

A binary decision variable $x_i \in \{0, 1\}$ typically means **whether**, e.g. whether to carry out some activity i.

“How many” versus “whether”

In LP and IP, a decision variable x_i often means **how many** of some quantity, e.g. how many of product i should we manufacture?

A binary decision variable $x_i \in \{0, 1\}$ typically means **whether**, e.g. whether to carry out some activity i.

Notice set notation $\{0, 1\}$, not the real interval $[0, 1]$.

Recreation centre problem

We run a recreation centre, and we have some money to invest in new facilities. Want to maximise daily **usage**.

- Resource constraints: €120,000 budget; 12 acres of land.
- Selection constraint: stakeholders say we could build **either** swimming pool **or** tennis courts, not both.

Recreation facility	Expected usage (people/day)	Cost (Euros)	Land requirement (acres)
Swimming pool	300	35000	4
Tennis centre	90	10000	2
Athletic centre	400	25000	7
Gym	150	90000	3

Recreation centre: model

- x_i means **whether** to build facility i
- 1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym

Maximise $300x_1 + 90x_2 + 400x_3 + 150x_4$
Subject to:

$$\begin{aligned} 35,000x_1 + 10,000x_2 + 25,000x_3 + 90,000x_4 &\leq 120,000 \\ 4x_1 + 2x_2 + 7x_3 + 3x_4 &\leq 12 \text{ acres} \\ x_1 + x_2 &\leq 1 \text{ facility} \end{aligned}$$

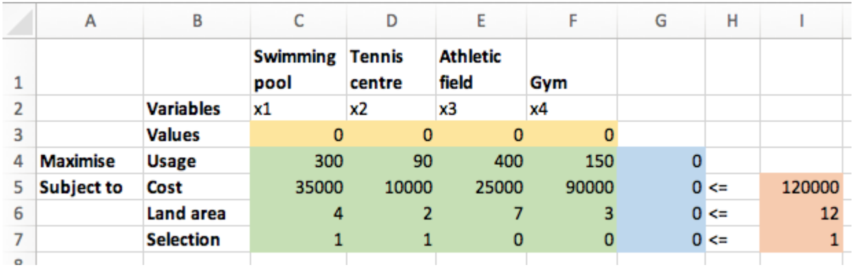
Binary variables and logical constraints

In the recreation centre model we had to write a **logical** constraint: we could build the Swimming Pool OR Tennis Centre (or neither) but not both.

We used **binary variables** to enforce this: $x_1 + x_2 \leq 1$.

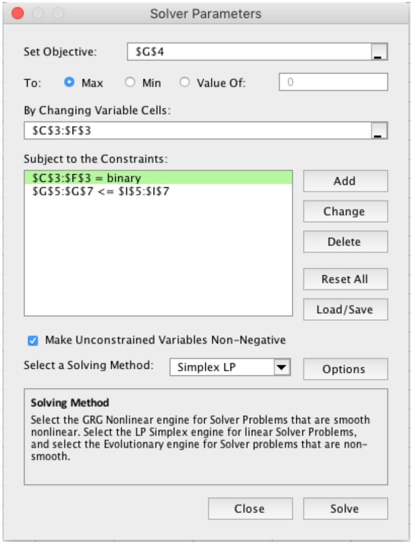
Recreation centre solution

We can program the model in Excel…



	A	B	C	D	E	F	G	H	I
1			Swimming pool	Tennis centre	Athletic field	Gym			
2		Variables	x1	x2	x3	x4			
3		Values	0	0	0	0			
4	Maximise	Usage	300	90	400	150	0		
5	Subject to	Cost	35000	10000	25000	90000	0	<=	120000
6		Land area	4	2	7	3	0	<=	12
7		Selection	1	1	0	0	0	<=	1

Recreation centre solution


Solver Parameters

Set Objective: $G$4

To: [✓ Max] Min Value Of: 0

By Changing Variable Cells:
$C$3:$F$3

Subject to the Constraints:
$C$3:$F$3 = binary
$G$5:$G$7 <= $I$5:$I$7

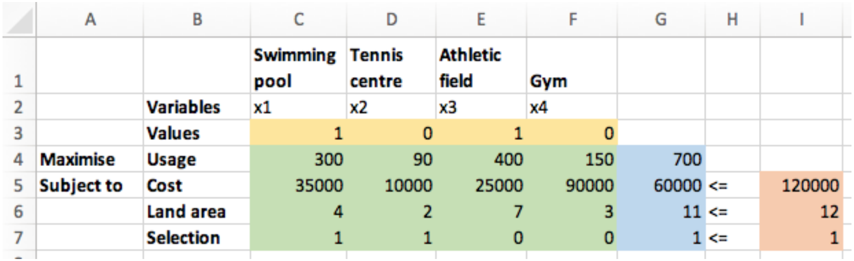
[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

Solving Method
Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

… and solve using a plug-in called `Solver` …

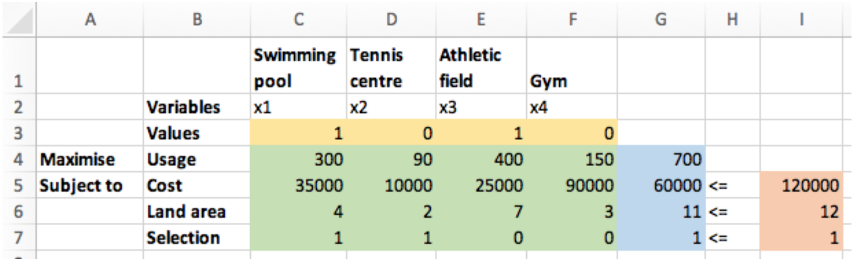
Recreation centre solution



	A	B	C	D	E	F	G	H	I
1			Swimming pool	Tennis centre	Athletic field	Gym			
2		Variables	x1	x2	x3	x4			
3		Values	1	0	1	0			
4	Maximise	Usage	300	90	400	150	700		
5	Subject to	Cost	35000	10000	25000	90000	60000	<=	120000
6		Land area	4	2	7	3	11	<=	12
7		Selection	1	1	0	0	1	<=	1

It finds $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$ for total usage of 700.

Recreation centre solution



	A	B	C	D	E	F	G	H	I
1			Swimming pool	Tennis centre	Athletic field	Gym			
2		Variables	x1	x2	x3	x4			
3		Values	1	0	1	0			
4	Maximise	Usage	300	90	400	150	700		
5	Subject to	Cost	35000	10000	25000	90000	60000	<=	120000
6		Land area	4	2	7	3	11	<=	12
7		Selection	1	1	0	0	1	<=	1

It finds $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$ for total usage of 700.

(We'll cover Excel Solver in labs.)

A weird language for logical constraints

When modelling problems, especially in BIP, we often need to write logical constraints. But LP/IP requires our constraints to be linear equations/inequalities.

Recall we wrote “Swimming or Tennis or neither but not both” as: $x_1 + x_2 \leq 1$.

(1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym)

A weird language for logical constraints

(1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym)

How could we express each of these conditions in binary variables?

- Exactly one of Athletic and Gym
- No more than 3 in total
- If Swimming then also Tennis

A weird language for logical constraints

(1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym)

How could we express each of these conditions in binary variables?

- Exactly one of Athletic and Gym
- No more than 3 in total
- If Swimming then also Tennis

- Exactly one of Athletic and Gym: $x_3 + x_4 = 1$

A weird language for logical constraints

(1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym)

How could we express each of these conditions in binary variables?

- Exactly one of Athletic and Gym
- No more than 3 in total
- If Swimming then also Tennis

- Exactly one of Athletic and Gym: $x_3 + x_4 = 1$
- No more than 3 in total: $x_1 + x_2 + x_3 + x_4 \leq 3$

A weird language for logical constraints

(1: Swimming pool, 2: Tennis centre, 3: Athletic field, 4: Gym)

How could we express each of these conditions in binary variables?

- Exactly one of Athletic and Gym
- No more than 3 in total
- If Swimming then also Tennis

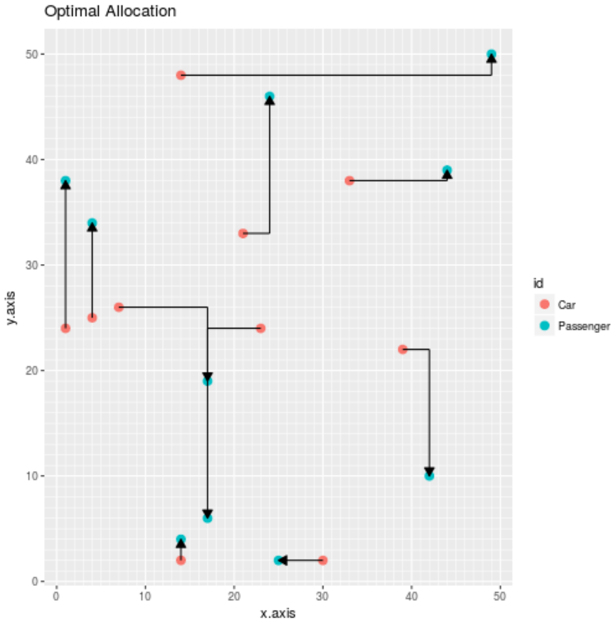
- Exactly one of Athletic and Gym: $x_3 + x_4 = 1$
- No more than 3 in total: $x_1 + x_2 + x_3 + x_4 \leq 3$
- If Swimming then also Tennis: $x_2 \geq x_1$

More logical constraints

In fact we can express many logical constraints:

Logical constraint	Implementation
Exactly 1 of A, B, C, D	$a + b + c + d = 1$
At most N of A, B, C, D	$a + b + c + d \leq N$
If A then B	$b \geq a$
If A then not B	$a + b \leq 1$
If not A then B	$a + b \geq 1$
If A then B, and if B then A	$a = b$
If A then B and C	$b \geq a, c \geq a$
If A then B or C	$b + c \geq a$
If B or C then A	$a \geq 0.5 * (b + c)$
If B and C then A	$a \geq b + c - 1$

Uber: assign cars to passengers



Uber: assign cars to passengers

We have a set of cars and a set of passengers. For simplicity we'll assume we have **enough** cars. We want to decide which car should pick up which passenger, minimising overall wait time.

Uber: assign cars to passengers

Data needed: travel time from each car to each passenger. We have several options:

- We could say travel time is proportional to distance, e.g. Euclidean distance
- In this picture, it looks like another common definition of distance might be better: Manhattan distance, also known as taxi-driver's distance:

$$d_M(x, y) = \sum_i |x_i - y_i|$$

- We could use a geographical information system with route-finding and live traffic information, e.g. Google Maps.

This gives us a matrix of car-passenger travel times T.

Uber: assign cars to passengers

- Decision variables (binary, double-subscripted) say **whether** car *i* picks up passenger *j*:

$$x_{ij} \in \{0, 1\}$$

- Cost function:

$$f(x) = \sum_{i,j} x_{ij} T_{ij}$$

- No car *i* can pick up **more than 1** passenger:

$$\sum_{j} x_{ij} \leq 1, \ \forall \ i$$

- Every passenger *j* must be picked up by **exactly** 1 car:

$$\sum_{i} x_{ij} = 1, \ \forall \ j$$

Assignment problems

The Uber problem is an **assignment problem**, meaning you have two sets of objects and you have to assign each item in one set to one or more items in the other set.

Another example is a scenario where we have several employees, each with differing skill-sets. We have several tasks, each requiring a certain number of person-hours and a certain set of skills. Which employee should work on which task?

Overview

Committee problem

NUI Galway are forming a review committee to consist of: at least one male, one female, one student, one academic staff and one administration staff. (One person could fulfill more than one requirement.)

Ten individuals have been nominated called 1, 2, …10. All participants will be paid equal expenses. Formulate the ILP to find the minimum cost committee that satisfies the requirements.

Individual	Sub-group
1, 2, 3, 4, 5	Female
6, 7, 8, 9, 10	Male
1, 2, 3, 10	Student
5, 6	Admin
4, 7, 8, 9	Academic

Committee problem

Binary decision variables x_i “whether member i is in the committee”.

Objective: minimise number of members =

$$\sum_i x_i$$

But how to write the constraints?

Committee problem

Rewrite the sub-groups as binary parameters s_{ij}: each person i is either in sub-group j or not.

NB s_{ij} are **data**, not **variables**. E.g. person 1 is female, so $s_{11} = 1$.

Individual	Sub-group	s_{ij}	j
1, 2, 3, 4, 5	Female	1111100000	1
6, 7, 8, 9, 10	Male	0000011111	2
1, 2, 3, 10	Student	1110000001	3
5, 6	Admin	0000110000	4
4, 7, 8, 9	Academic	0001001110	5

Now write one constraint for each sub-group j:

$$\sum_i x_i s_{ij} \geq 1$$

Multi-period scheduling

A computer manufacturer has this **order schedule**:

Week	0	1	2	3	4	5
Computer Orders	105	170	230	180	150	250

- Production capacity: 160 computers/week regular time, 50 extra with overtime.
- Assembly Costs: €190/computer regular time; €260/computer overtime.
- Inventory Cost: €10/computer per week.

Goal: determine a **production schedule** to meet orders at minimum cost with no left over inventory at end of this production period.

Multi-period scheduling: formulation

Define decision variables:

- r_j = regular production of computers per week j $(0\ldots5)$
- o_j = overtime production of computers per week j $(0\ldots5))$
- i_j = extra computers carried forward as inventory from week j $(0\ldots5)$

This makes it easy to write:

- the objective (minimize cost) and;
- constraints:
  - each week, the inventory carried in + that week's production - that week's order = inventory carried forward
  - (notice inventory carried in to week 0 is 0!)
  - at the end, inventory = 0.

California Manufacturing Co.

The California Manufacturing Company is considering expansion by building a new factory in Los Angeles or San Francisco, or both. It is also considering building at most one new warehouse, but the choice of warehouse location is restricted to a city where a new factory is built. The total investment budget is \$10M. The net present values (NPV) for each alternative and some other information are shown below. Maximise the total NPV using an ILP. (From Hiller and Lieberman.)

Question	DV	NPV	Capital Requirement
LA Factory?	x_1	\$9M	\$6M
LA Warehouse?	x_2	\$5M	\$3M
SF Factory?	x_3	\$6M	\$5M
SF Warehouse?	x_4	\$4M	\$2M

California Manufacturing Co.: model

- Decision variables: x_i: whether to build facility i.
- Objective: maximise NPV = $9x_1 + 5x_2 + 6x_3 + 4x_4$
- Constraints:
    - 10 Million to invest: $6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10$
    - At least one factory: $x_1 + x_2 \geq 1$
    - At most one warehouse: $x_3 + x_4 \leq 1$
    - LA – if warehouse then factory: $x_1 \geq x_3$
    - SF – if warehouse then factory: $x_2 \geq x_4$

(E.g. for LA, the case that is disallowed is $x_1 = 0$ and $x_3 = 1$.)

Capital Budgeting Problem (BIP)

A company is planning its capital spending for the next T periods. There are N projects that compete for the limited capital B_j, available for investment in period j. Each project requires a certain amount of investment in each period once it is selected. Let a_{ij} be the required investment in project i for period j. v_i is the Net Present value (think: profit) of project i. The problem is to select the projects for investment that will maximise the net present value of the projects selected.

Let $x_i = 1$ indicate we DO invest in project i. Objective: maximise $\sum_i v_i x_i$ subject to $\sum_i a_{ij} x_i \leq B_j, \ \forall j \in 1 \dots T$. Also subject to $x_i \in \{0, 1\}$.

Capital Budgeting Problem (BIP)

	Expenditures (in Millions of €)			
Project	Year 1	Year 2	Year 3	NPV
1	5	1	8	20
2	4	7	10	40
3	3	9	2	20
4	7	4	1	15
5	8	6	10	30
Funds Available	25	25	25	

Then this problem can be expressed as:

Maximise $z = 20x_1 + 40\, x_2 + 20\, x_3 + 15\, x_4 + 30\, x_5$

Subject to:

$$5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 \leq 25$$
$$1x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \leq 25$$
$$8x_1 + 10x_2 + 2x_3 + 1x_4 + 10x_5 \leq 25$$

Capital Budgeting Problem (BIP)

Suppose we solve the LP relaxation. We get $z = 125$, $x = (0, 2.27, 0, 2.27, 0)$. What does this tell us?

Capital Budgeting Problem (BIP)

Suppose we solve the LP relaxation. We get $z = 125$, $x = (0, 2.27, 0, 2.27, 0)$. What does this tell us?

This gives an **upper bound** on profit, but it is very unrealistic!

Capital Budgeting Problem (BIP)

Suppose we solve the LP relaxation. We get $z = 125$, $x = (0, 2.27, 0, 2.27, 0)$. What does this tell us?

This gives an **upper bound** on profit, but it is very unrealistic!

More realistic: instead of dropping the integer constraint $x_i \in \{0, 1\}$, **convert** it to $x_i \in [0, 1]$. This is **more** realistic. Often this is what people mean when they say LP relaxation.

Now we get a **lower** upper bound $z = 109$, $x = (0.58, 1, 1, 1, 0.74)$.

Capital Budgeting Problem (BIP)

Suppose we solve the LP relaxation. We get $z = 125$, $x = (0, 2.27, 0, 2.27, 0)$. What does this tell us?

This gives an **upper bound** on profit, but it is very unrealistic!

More realistic: instead of dropping the integer constraint $x_i \in \{0, 1\}$, **convert** it to $x_i \in [0, 1]$. This is **more** realistic. Often this is what people mean when they say LP relaxation.

Now we get a **lower** upper bound $z = 109$, $x = (0.58, 1, 1, 1, 0.74)$.

But with the true (binary) constraints we get a lower value still, $z = 95$, $x = (1, 1, 1, 1, 0)$.

Constraint Programming

Constraint programming is a special case of LP/IP where there is no objective function – just constraints. Any solution that obeys all constraints is good enough! Usually that's because finding valid solutions is so hard in itself. In practice, e.g. exam timetabling may be a constraint programming problem.

Specialised algorithms are used, which we won't cover, but they are mentioned in *CT5137: Knowledge Representation & Statistical Relational Learning*.

Reading

Under "Integer Programming" from Beasley's OR-Notes http://people.brunel.ac.uk/~mastjjb/jeb/or/ip.html, read:

- Branch and bound algorithm (to be covered Lecture 04)
- Facility location
- Vehicle routing

Next week

Algorithms, software, and sensitivity.