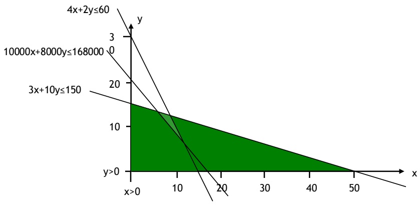
Linear Programming

Linear programming in low dimensions



Example

You can build two types of houses: X and Y.

X requires:
- 10.000 bricks
- 4 doors
- 3 windows — Price: $200.000

Y requires:
- 8.000 bricks
- 2 doors
- 10 windows — Price: $250.000

Example

X requires: 10.000 bricks
4 doors
3 windows Price: \$200.000

Y requires: 8.000 bricks
2 doors
10 windows Price: \$250.000

You have: 168.000 bricks
60 doors
150 windows

Maximise profit!
How many houses of type X and Y should be built?

Example

Set |X| = x and |Y|=y

Max objective function

$$f(x,y) = 200.000x + 250.000y$$

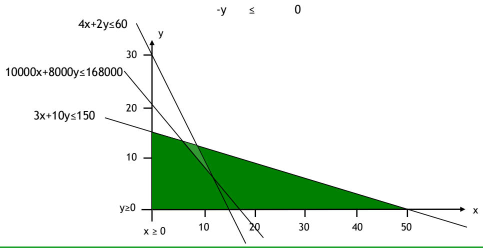
Constraints:

10.000x +	8.000y	≤	168.000 bricks	
4x +	2y	≤	60 doors	
3x +	10y	≤	150 windows	
-x		≤	0	(not a negative number of X houses)
	-y	≤	0	(not a negative number of Y houses)

Linear optimisation problem (in 2 dimensions)!

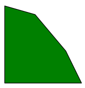
Geometric interpretation

Max	f(x,y) = 200.000x + 250.000y			
Constraints:	10.000x +	8.000y	≤	168.000 bricks
	4x +	2y	≤	60 doors
	3x +	10y	≤	150 windows
	-x		≤	0
		-y	≤	0



Geometric interpretation

Max	f(x,y) = 200.000x + 250.000y			
Constraints:	10.000x + 8.000y		≤	168.000 bricks
	4x +	2y	≤	60 doors
	3x +	10y	≤	150 windows
	-x		≤	0
		-y	≤	0

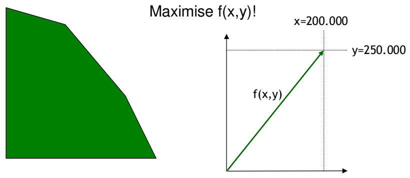


Geometric interpretation

Max	f(x,y) = 200.000x + 250.000y			
Constraints:	10.000x + 8.000y		≤	168.000 bricks
	4x +	2y	≤	60 doors
	3x +	10y	≤	150 windows
	-x		≤	0
		-y	≤	0

How do I find the optimal solution?

Maximise f(x,y)!

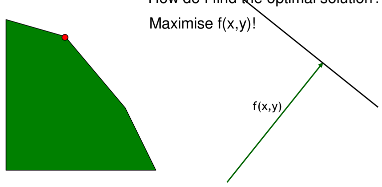


Geometric interpretation

Max	f(x,y) = 200.000x + 250.000y			
Constraints:	10.000x + 8.000y		≤	168.000 bricks
	4x +	2y	≤	60 doors
	3x +	10y	≤	150 windows
	-x		≤	0
		-y	≤	0

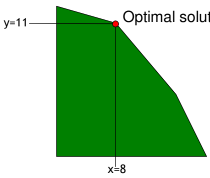
How do I find the optimal solution?

Maximise f(x,y)!



Geometric interpretation

Max	$f(x,y) = 200.000x + 250.000y$		
Constraints:	$10.000x + 8.000y$	$\le$	168.000 bricks
	$4x + 2y$	$\le$	60 doors
	$3x + 10y$	$\le$	150 windows
	$-x$	$\le$	0
	$-y$	$\le$	0



$\Rightarrow$ 8 houses of type X and 11 houses of type Y

Linear programming: general case

Optimal solution x=8 and y=11.

- x and y are integers…luckily!
- if we add integer constraints $\Rightarrow$ integer linear program which is NP-hard.
- consider only linear constraints ("linear programming")

Linear programming: general case

Maximise objective function

$$f(x_1, x_2, \ldots, x_d) = c_1x_1 + c_2x_2 + \ldots + c_dx_d$$

Constraints:

$$\begin{array}{llllll} a_{11}x_1 & + a_{12}x_2 & + & \ldots & + a_{1d}x_d & \le b_1 \\ a_{21}x_1 & + a_{22}x_2 & + & \ldots & + a_{2d}x_d & \le b_2 \\ a_{31}x_1 & + a_{32}x_2 & + & \ldots & + a_{3d}x_d & \le b_3 \\ & & & & & \\ a_{n1}x_1 & + a_{n2}x_2 & + & \ldots & + a_{nd}x_d & \le b_n \end{array}$$

Linear optimisation problem in d dimensions with n constraints.

The imagination driving Australia's ICT future.

Geometric interpretation

- Each constraint represents a half space in R^d.
- The intersection (feasible region) of all the half spaces is the set of feasible solutions.
- The intersection is a convex polyhedron!
  May be unbounded.

The imagination driving Australia's ICT future.

Geometric interpretation

- How can we solve a linear program (LP)?
  - simplex method (Dantzig, 1947) - exponential time
  - ellipsoid method (Khachiyan, 1972) - $O(d^4L)$ (L = #bits in input)
  - interior point method (Karmarkar, 1984) - $O(d^{3.5}L)$
- Integer Linear Program is NP-hard
- Why computational geometry?
  Gives faster algorithms in low dimensions.
  Meggido 1984, Clarkson, Dyer 1986 $O(3^{d^2} \cdot n)$ time.
  Seidel $O(d! \cdot n)$ expected running time, 1991.

The imagination driving Australia's ICT future.

Geometric interpretation

We want to maximise the objective function

$$f(x_1,x_2, \ldots, x_d) = c_1x_1 + c_2x_2 + \ldots + c_dx_d$$

⇔ Find a point in a convex polyhedron that is extreme in direction

$$c=(c_1,c_2, \ldots , c_d)$$

In 2D we have:

-the vector defining the objective function is $c=(c_x,c_y)$
-the objective function is $f_c(p)=c_xp_x+c_yp_y$
-a set of n linear constraints $\cap H=h_1 \cap h_2 \cap \ldots \cap h_n$

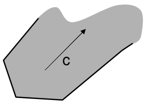
Find a point $p \in \cap H$ such that $f_c(p)$ is maximised!

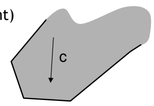
Denote this LP by (H,c).

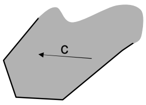
Geometric interpretation

Four cases:

1. Infeasible (intersection of the half planes is empty)
2. Unbounded in direction c (return a ray)
3. Non-unique solution (report one point)
4. Unique solution





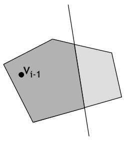


Incremental algorithm

Add constraints: $m_1 := p_x \leq M$
$m_2 := p_y \leq M$, for some large M. [bounds space]

LP=(H,c) $H_i = \{m_1, m_2, h_1, h_2, \dots, h_i\}$
$F_i = m_1 \cap m_2 \cap h_1 \cap h_2 \cap \dots \cap h_i$.

Observation: $F_0 \supseteq F_1 \supseteq F_2 \supseteq \dots \supseteq F_n$



Let v_i be a unique optimal solution in F_i.

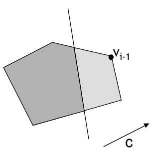
Lemma:
1. if $v_{i-1} \in h_i$ then $v_i = v_{i-1}$
2. if $v_{i-1} \notin h_i$ then $F_i = \emptyset$ or v_i lies on the boundary of h_i.

Incremental algorithm

Add constraints: $m_1 := p_x \leq M$
$m_2 := p_y \leq M$, for some large M. [bounds space]

LP=(H,c) $H_i = \{m_1, m_2, h_1, h_2, \dots, h_i\}$
$F_i = m_1 \cap m_2 \cap h_1 \cap h_2 \cap \dots \cap h_i$.

Observation: $F_0 \supseteq F_1 \supseteq F_2 \supseteq \dots \supseteq F_n$



Let v_i be a unique optimal solution in F_i.

Lemma:
1. if $v_{i-1} \in h_i$ then $v_i = v_{i-1}$ [follows from Observation]
2. if $v_{i-1} \notin h_i$ then $F_i = \emptyset$ or v_i lies on the boundary of h_i.
   [Proof: Assume the opposite…]

The imagination driving Australia's ICT future.

Incremental algorithm

Observation: In step i the optimal point v_i can be found in O(i) time.

Algorithm(H,c):

1. let v_0 be the corner corresponding to c of $m_1 \cap m_2$
2. for i=1 to n do
3. if $v_{i-1} \in h_i$
4. then $v_i = v_{i-1}$
5. else set v_i to be the point on the boundary of h_i that maximises $f_c(p)$
6. if no such point exists then report LP infeasible
7. return v_n

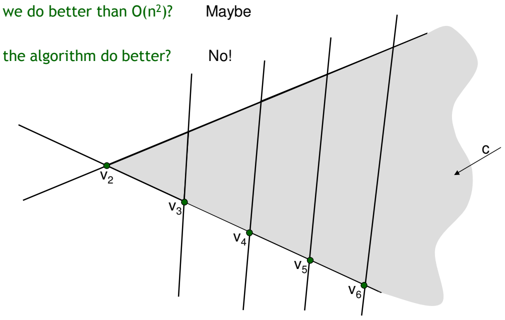
Running time: $\sum_{i=1}^{n} O(i) = O(n^2)$. Space: O(n)

The imagination driving Australia's ICT future.

Incremental algorithm

Can we do better than $O(n^2)$? Maybe

Can the algorithm do better? No!

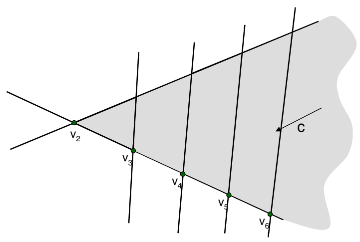


The imagination driving Australia's ICT future.

Incremental algorithm

What if we added the half planes in the following order: $h_1, h_2, h_n, h_{n-1}, \ldots, h_3$

Running time: O(n)



The imagination driving Australia's ICT future.

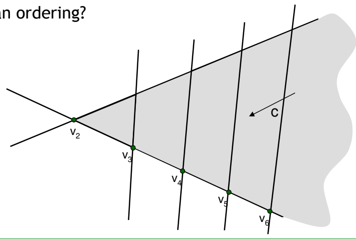
NATIONAL ICT AUSTRALIA

Incremental algorithm

Is there always an ordering that guarantees a running time of O(n)?

Can we always find such an ordering?

Random order!



22

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Random incremental algorithm

The running time depends on the ordering,
and the number of possible orderings is n!

Lemma: The randomised algorithm requires O(n) expected time.

Proof:

Adding a half plane requires: O(1) time if $v_{i-1} \in h_i$

O(i) time if $v_{i-1} \notin h_i$

Let X_i be a random variable : 0 if $v_{i-1} \in h_i$

1 if $v_{i-1} \notin h_i$

Total time: $n + \Sigma\, O(i) \cdot X_i$

Use linearity of expectation! $E[\, \Sigma\, O(i) \cdot X_i \,] = \Sigma\, O(i) \cdot E[X_i]$

(the expected value of the sum is the sum of the expected values)

23

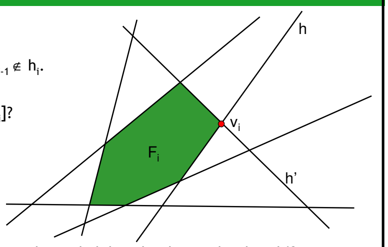
The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Random incremental algorithm

What is $E[X_i]$?
It is the probability that $v_{i-1} \notin h_i$.

How can we calculate $E[X_i]$?



Remove h_i to get F_{i-1}. What is the probability that h_i is either h or h'?

$E[X_i] \leq 2/i$

Total expected time: $\Sigma\, O(i) \cdot 2/i = O(n)$

24

NATIONAL ICT AUSTRALIA

Summary

Maximise objective function: $f(x_1, x_2, \ldots, x_d) = c_1x_1 + c_2x_2 + \ldots + c_dx_d$

Constraints:

$$
\begin{array}{llllll}
a_{11}x_1 & + a_{12}x_2 & + & \ldots + a_{1d}x_d & \leq & b_1 \\
a_{21}x_1 & + a_{22}x_2 & + & \ldots + a_{2d}x_d & \leq & b_2 \\
a_{31}x_1 & + a_{32}x_2 & + & \ldots + a_{3d}x_d & \leq & b_3 \\
a_{n1}x_1 & + a_{n2}x_2 & + & \ldots + a_{nd}x_d & \leq & b_n
\end{array}
$$

Linear optimisation problem in d dimensions with n constraints.

2D: Incremental algorithm with $O(n^2)$ running time