Devoir 1

roman nakhuda 20207082
lucky khounvongsa 20172476
Ange lilian tchomtchoua tokam 20210129

optimisation lineaire - IFT2505

Professeur fabien bastin

Université de Montréal

À remettre le 2 Octobre 2023 à 10:00

Question 1

En prenant x_1 comme le nombre d'acres de tomates et x_2 comme le nombre d'acres de maïs,

On obtient le probleme mathematique suivant;

$$\begin{aligned} \max \ & 30x_1 + 15x_2 \\ \text{sujet à } & x_1 + x_2 \leq 350 \qquad \text{:pour les acres} \\ & 3x_1 + x_2 \leq 480 \qquad \text{:pour les heures} \\ & x_1, x_2 \geq 0 \end{aligned}$$

Que l'on peut convertir en un problème de minimisation

à l'introduction des variables d'écart u et v, on a;

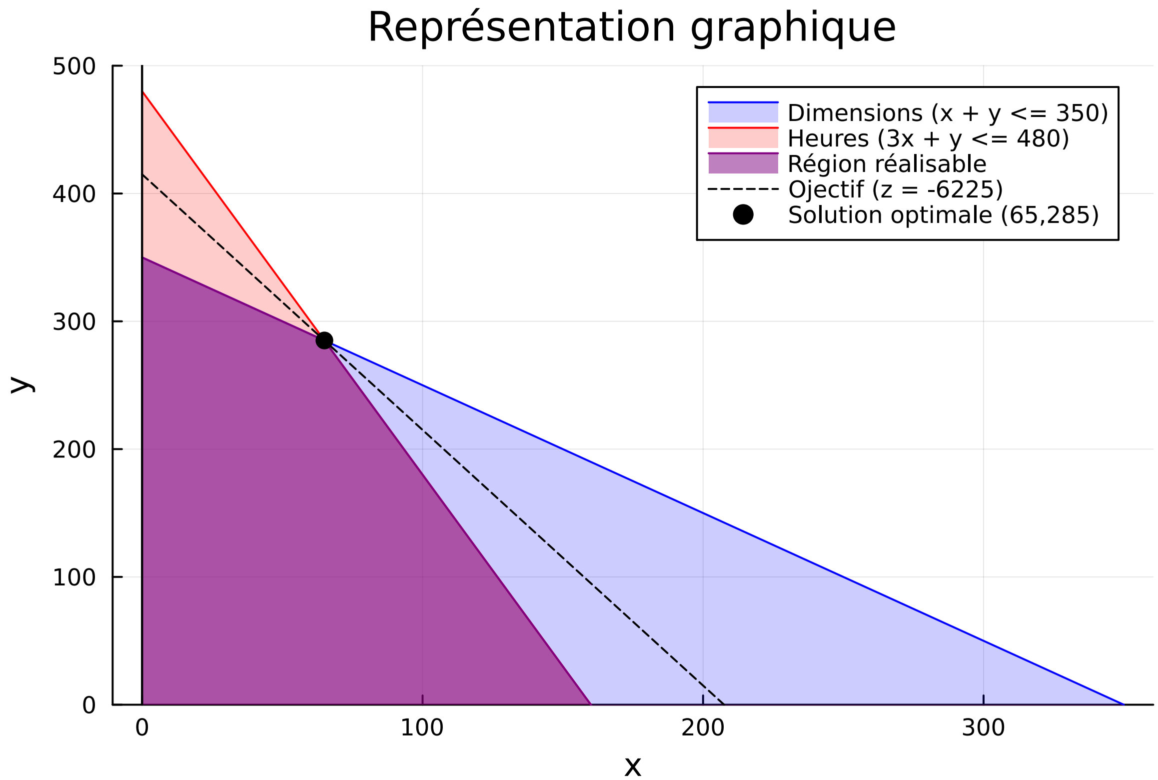
$$\begin{aligned} -min \ & -30x_1 - 15x_2 \\ \text{sujet à } & x_1 + x_2 + u = 350 \\ & 3x_1 + x_2 + v = 480 \\ & x_1, x_2, u, v \geq 0 \end{aligned}$$

- **Resolution sous forme Graphique**

In [1]:
```
using JuMP, HiGHS, Plots
```

In [2]:
```
#Premiere contrainte
plot(x-> 350-x, 0, 350, fill = (0, 0.2, :blue), color = :blue, label = "Dimensions
#Deuxieme contrainte
plot!(x-> 480-3x, 0, 480/3, fill = (0, 0.2, :red), color = :red, label = "Heures (3
plot!(x-> min(350-x, 480-3x), 0, 350,
    fill = (0, 0.5, :purple), color = :purple, label = "Région réalisable")
vline!([0,0], color = :black, label = "")
plot!(ylims = (0, 500))
plot!(x -> (6225 - 30*x)/15, color = :black, label = "Ojectif (z = -6225)", linesty
scatter!([65], [285], label = "Solution optimale (65,285)", color = :black, markers

plot!(xlabel = "x", ylabel = "y")
plot!(title = "Représentation graphique")
```



D'où la solution optimale en $x = 65$ et $y = 285$, **avec** $z = 6225$

- **Resolution avec les librairies JuMP et HiGHS**

In [3]:
```
using HiGHS, JuMP
```

In [4]:
```
m = Model(HiGHS.Optimizer)

@variable(m, x >= 0)
@variable(m, y >= 0)

@constraint(m, x + y <= 350)
@constraint(m, 3x + y <= 480)

@objective(m, Max, 30x+15y);

print(m)
```

```
Max 30 x + 15 y
Subject to
 x + y <= 350
 3 x + y <= 480
 x >= 0
 y >= 0
```

In [5]:
```
status = optimize!(m)
```

```
Running HiGHS 1.5.3 [date: 1970-01-01, git hash: 45a127b78]
Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
2 rows, 2 cols, 4 nonzeros
2 rows, 2 cols, 4 nonzeros
Presolve : Reductions: rows 2(-0); columns 2(-0); elements 4(-0) - Not reduced
Problem not reduced by presolve: solving the LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0    -4.4999962776e+01 Ph1: 2(6); Du: 2(45) 0s
          2     6.2250000000e+03 Pr: 0(0) 0s
Model   status      : Optimal
Simplex   iterations: 2
Objective value     :  6.2250000000e+03
HiGHS run time      :          0.00
```

In [6]:
```julia
[value(x), value(y)]
```

```
2-element Vector{Float64}:
  65.0
 285.0
```

- **Resolution avec Julia grace à la fonction du simplexe**

In [7]:
```julia
function pivot!(M::Matrix, i::Int, j::Int)
    m, n = size(M)
    @assert M[i, j] != 0
    M[i, :] = M[i, :]/M[i, j]
    for k in setdiff(1:m, i)
        M[k, :] -= M[k, j] * M[i, :]
    end
    return M
end

function getReducedCosts(M::Matrix)
    m, n = size(M)
    return M[end, 1:n-1]
end
function getxB(M::Matrix)
    m, n = size(M)
    return M[1:m-1, end]
end
function enteringVar(M::Matrix)
    rc = getReducedCosts(M)
    index = argmin(rc)
    return rc[index] >= 0 ? -1 : index
end
function exitingVarIndex(M::Matrix{T}, enteringVar::Int) where T
    col = M[1:end-1, enteringVar]
    xB = getxB(M)
    m, n = size(M)
    index = -1
    val = T(Inf)
    for i in 1:m-1
        if (col[i] > 0) && (xB[i]/col[i] < val)
```

```julia
            val = xB[i]/col[i]
            index = i
        end
    end
    return index
end
function isOneHot(v::Vector)
    n = length(v)
    return (sum(iszero, v) == n-1) && (sum(isone, v) == 1)
end
function isoptimal(M)
    return enteringVar(M) == -1
end
function findInitialBasis!(M::Matrix)
    m, n = size(M)
    m-=1
    n-=1
    basis = [-1 for _ in 1:m]
    for i in 1:n
        if isOneHot(M[1:end-1, i])
            index = findfirst(isone, M[:, i])
            basis[index] = i
        end
    end
    @assert !any(t-> t == -1, basis) "problem not caconical"
    for i in 1:m
        j = basis[i]
        pivot!(M, i, j)
    end
    return basis
end
function simplexSolver(A::Matrix{T}, b::Vector, c::Vector; verbose::Bool = false) w
    M = [A b; c' 0]
    basis = findInitialBasis!(M)
    k = 1
    nmax = 1000
    while !isoptimal(M) && k < nmax
        k+=1
        verbose && display(M)
        entering = enteringVar(M)
        entering == -1 && (println("------------"); return T[-1, -1])
        leaving = exitingVarIndex(M, entering)
        leaving == -1 && (println("------------"); return T[-1, -1])
        verbose && @show (entering, leaving)
        basis[leaving] = entering
        pivot!(M, leaving, entering)
    end
    verbose && display(M)
    m, n = size(M)
    xstar = zeros(T, n - 1)
    xstar[basis] = getxB(M)
    return xstar
end
```

```
simplexSolver (generic function with 1 method)
```

In [8]:
```julia
A=[1//1 1 1 0 ;
    3 1 0 1;
]
b=[350, 480]
c = [-30, -15,0,0]
```

```
4-element Vector{Int64}:
 -30
 -15
   0
   0
```

In [9]:
```julia
simplexSolver(A, b, c, verbose=true)
```

```
3×5 Matrix{Rational{Int64}}:
   1//1    1//1  1//1  0//1  350//1
   3//1    1//1  0//1  1//1  480//1
 -30//1  -15//1  0//1  0//1    0//1
3×5 Matrix{Rational{Int64}}:
 0//1   2//3  1//1  -1//3   190//1
 1//1   1//3  0//1   1//3   160//1
 0//1  -5//1  0//1  10//1  4800//1
3×5 Matrix{Rational{Int64}}:
 0//1  1//1   3//2  -1//2   285//1
 1//1  0//1  -1//2   1//2    65//1
 0//1  0//1  15//2  15//2  6225//1
(entering, leaving) = (1, 2)
(entering, leaving) = (2, 1)
4-element Vector{Rational{Int64}}:
  65//1
 285//1
   0//1
   0//1
```

Question 2

Soit le problème écrit au long :

$$\begin{aligned} \min\ & c_1x_1 + \cdots + c_nx_n \\ \text{tel que } & a_1x_1 + \cdots + a_nx_n \leq b \\ & x_1, \ldots, x_n \geq 0 \end{aligned}$$

On considère x_k **comme la variable de base du problème avec** $A = (a_1 \ldots a_k \ldots a_n)$.

Comme x_k **est l'unique variable de base du problème nous avons :**

$B = (a_k) \implies B^{-1} = \frac{1}{a_k}$

On a donc : $Ax = b \implies Bx_b = b \implies x_b = bB^{-1} \implies x_k = \frac{b}{a_k}$

$$O\text{ù} \qquad X = \begin{pmatrix} x_b & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ \frac{b}{a_k} \\ \vdots \\ 0 \end{pmatrix}$$

C'est une solution de base non dégénérée puisque $b > 0$ **et** $a_k > 0$ **(1)**

On a que l'indice k satisfait la relation $\frac{c_k}{a_k} = \min\limits_{j=1,\ldots,n} \{\frac{c_j}{a_j}\}$

par le fait que $\frac{c_k}{a_k}$ **est le** $\min\limits_{j=1,\ldots,n} \{\frac{c_j}{a_j}\}$ **et par (1) on a ainsi :**

$z = c_k \frac{b}{a_k} = \frac{c_k}{a_k} * b$ **alors**

$$X = \begin{pmatrix} 0 \\ \vdots \\ \frac{b}{a_k} \\ \vdots \\ 0 \end{pmatrix}$$

Puisque b
est constant et qu'il apparaît pour toute solutions de base non dégénérée,

on peut se concentrer uniquement sur la valeur de $\frac{c_j}{a_j}$.
afin d'évaluer la fonction objective.

Donc comme $\frac{c_k}{a_k}$ est le $\min\limits_{j=1,...,n}\{\frac{c_j}{a_j}\}$ Alors la solution est optimale pour $\frac{c_k}{a_k}$.

X est bien une solution de base optimale non dégénérée puisque z est minimisé lorsqu'on choisi x_k comme variable de base.

Par le théorème fondamentale de la programmation linéaire,

c'est aussi une solution optimale du problème.

Question 3

Soit $K = \{x|Ax = b, x \geq 0\}$ **K est non vide**

Corollaire 1:

Montrons que si l'ensemble convexe K est non vide, alors il existe au moins un

Par définition d'un ensemble convexe non vide, chaque $x \in k$
peut s'écrire comme combinaison convexe de deux points $x_1, x_2 \in k$
tels que $x = \alpha x_1 + (1-\alpha)x_2, \alpha \in [0, 1]$

Par définition d'un point extrême, chaque x est un point extrême de k s'
il n'existe pas de deux points $x_1, x_2 \in k$ **tels que** $x = \alpha x_1 + (1-\alpha)x_1, \alpha \in]0, 1[$

Notons S **l'ensemble des points qui respectent;**

$x = \alpha x_1 + (1-\alpha)x_2$**, avec** $x_1, x_2 \in k$ **et** $\alpha \in]0, 1[$

Ainsi, $S \subseteq k$ **, or** $k \setminus S \neq \emptyset$ **, car** ∃x ∈k **tel que** $x = \alpha x_1 + (1-\alpha)x_2$ **, pour**
$x_1, x_2 \in k$ **et** $\alpha = 0$ ou $\alpha = 1$

Ainsi lorsqu'on prend l'ensemble K
privé de tous les points qui ne sont pas des points extremes,
on aura un ensemble resultant qui est non-vide.

Par conséquent, si l'ensemble convexe K
est non-vide, alors il y a au moins un point extrême.

Corollaire 2:

Par le théorème de fondamentale de la programmation linéaire, nous savons q

s'il y a une solution réalisable optimale, alors il y a une solution de base réalisa

Ainsi, s'il existe une solution optimale finie à un problème,

alors il existe une solution optimale finie de base à ce problème.

On pose ainsi le vecteur x^* **dans** $Ax^* = b$ **et** $x^* \geq 0$,
une solution de base optimale de ce problème.

Or x^* **est une solution de base réalisable si et seulement si** x^*
est un point extrême.

Ainsi il existe bien une solution optimale finie qui est un point extrême.

Question 4

Soit

$$\min\ c^T x$$
$$\text{tel que } Ax = b, x \geq 0$$

a) Montrons que d direction réalisable $\implies Ad = 0$

$$\begin{aligned}
&A(x + \alpha d) = b \\
&Ax + A(\alpha d) = b && (d\acute{e}velopement) \\
&b + A(\alpha d) = b && (Ax = b) \\
&A(\alpha d) = 0 && (-b) \\
&Ad = 0 && (\div \alpha)
\end{aligned}$$

$$QED$$

b) Nous pouvons écrire l'inégalité pour les points réalisables

$$C^T x^* \leq C^T x$$

Il existe $\alpha > 0$ **tel que** $x = x^* + \alpha d$ **On sait que**

$$\begin{aligned}
Ax^* = b \implies &A(x - \alpha d) = b \\
&Ax - A\alpha d = b \\
&Ax = b
\end{aligned}$$

On a x donc réalisable, qui nous donne;

$$\begin{aligned}
\implies &C^T x^* \leq C^T x \\
&C^T x^* \leq C^T (x^* + \alpha d) && (ici, x^{'} = x^* + \alpha d) \\
&C^T x^* \leq C^T x^* + C^T \alpha d && (D\acute{e}velopement) \\
&0 \leq C^T \alpha d && (-C^T x^*) \\
&0 \leq C^T d && (\div \alpha)
\end{aligned}$$

$$QED$$