1 Introducció al paquet de programació lineal GLPK

En aquesta pràctica i la següent introduirem el GLPK (GNU Linear Programming Kit) i l'anirem usant de manera progressiva. La primera sessió servirà per aprendre a fer funcionar el GLPK: escriure un problema senzill en llenguatge GNU MathProg, cridar el glpsol des d'una consola, i interpretar-ne la sortida. Començarem amb uns quants exemples de guia.

Per a la instal·lació i detalls de funcionament vegeu les instruccions penjades a la intranet de l'assignatura.

1.1 Escriptura del model, execució del programa glpsol i lectura de la solució.

Considerem, per començar, el problema

$$\begin{aligned} &\text{maximitzar} && z = 2x + 3y \\ &\text{subjecte a:} && \begin{cases} x + 2y \leq 10 \\ 3x + 2y \leq 15 \\ y \leq 4 \\ x, y \geq 0 \end{cases} \end{aligned}$$

Com ho fem en un paper, quan escrivim el programa lineal en MathProg per fer córrer el glpsol, hem de declarar:

- Les variables: x i y, que són positives i reals.
- La funció objectiu: $2x + 3y$
- Les restriccions: $\begin{cases} x + 2y \leq 10 \\ 3x + 2y \leq 15 \\ y \leq 4 \end{cases}$

Introducció del model en un fitxer

Obrim un editor de fitxers, que permeti escriure sense format (Bloc de notas o Notepad en windows, o kate en kubuntu, o qualsevol altre que no afegeixi format). Hi escrivim les línies següents, on els números de línia no s'han d'escriure, hi són ara només per referir-nos a les diferents línies que hem escrit:

```
1 # Exemple 1
2 # Declaració de les variables
3 var x>=0;
4 var y>=0;
5
6 # Declaració de la funció objectiu
```

```
7  maximize z: 2*x+3*y;
8
9  # Declaració de les restriccions
10 subject to r1: x+2*y <= 10;
11 subject to r2: 3*x+2*y <= 15;
12 subject to r3: y <= 4;
13 end;
```

- La primera i la segona línies inclouen *comentaris*. Per indicar-ho, es comencen per `#`. Tot el que hi ha darrera d'aquest símbol no es té en compte en executar-se el glpsol. També són comentaris les línies 6 i 9. Els comentaris en un arxiu de codi ajuden a llegir-lo posteriorment, val la pena introduir-los per facilitar-nos la feina a l'hora de corregir errors, si cal, o fer petites modificacions.

- En les línies 3 i 4 *declarem les variables* del problema: x i y. Una declaració de variable comença amb `var`. També declarem alhora que són positives. Per defecte, les variables són reals, és a dir, no cal declarar que ho són.

- Observem que *totes les línies que no són comentaris acaben en punt i coma* (`;`). Deixar-se el punt i coma al final de línea és l'error que apareix més sovint quan compilem.

- La línea 7 *declara la funció objectiu*, $z = 2x + 3y$. En els programes lineals es pot *maximitzar* o *minimitzar* la funció objectiu, ho hem d'especificar començant la declaració de la funció objectiu amb `maximize` o `minimize`, respectivament. A continuació *hem de donar nom a la funció objectiu*, en aquest exemple li diem z. És molt important tenir en compte que:

  - S'usen els *dos punts* (`:`) *per separar el nom de la funció objectiu de la seva fórmula.*
  - L'asterisc (`*`) denota la multiplicació. Així com quan escrivim en matemàtiques no posar res indica producte, aquí és obligatori posar-hi asterisc. Aquest és un altre dels errors freqüents.

- En les línies 10, 11 i 12 hi *declarem les restriccions*. La primera restricció ha d'anar precedida de `subject to` o bé `s.t.`. Les altres restriccions, que es posen a continuació, no cal que vagin precedides de res. En aquest exemple, però, hem inclòs davant de cada restricció `subject to` per facilitar la lectura (vegeu exemple **??**, en que només apareix `subject to` davant la primera restricció només). En l'apartat de restriccions també és imprescindible donar un nom diferent a cada una de les restriccions, de la mateixa manera que ho hem fet amb la funció objectiu (per exemple, `r1: x+2*y <= 10`).

- Acabem el codi amb un `end;` a la línea 13.

- En general, a l'hora donar noms a les variables, la funció objectiu i les restriccions, *cal tenir en compte:*

  - han de ser diferents;
  - han de començar amb una lletra que pot ser majúscula o minúscula;
  - després de la primera lletra poden contenir altres lletres, números o barra baixa (`_`) però no altres caràcters com guió (`-`), admiració, coma, etc.;
  - poden ser tan llargs com vulguem;

– hem de respectar al llarg de tot el programa el nom exacte, tenint en compte que la *mateixa lletra en majúscula i en minúscula s'interpreta com a diferent.*

Ja hem acabat. Podem guardar-lo per exemple amb el nom de `ex1.mod` en una carpeta on poguem executar el glpsol.

Execució del programa glpsol

Obrim una finestra de comandes en la mateixa carpeta on hem guardat el fitxer `ex1.mod` i escrivim

```
glpsol -m ex1.mod -o ex1.sol
```

Aquesta línea de comandes usa les opcions següents:

- L'opció `-m` (seguida del nom `ex1.mod`) li diu al glpsol que el model està escrit al fitxer ex1.mod
- L'opció `-o` (seguida del nom `ex1.sol`) li diu al glpsol que escrigui la solució en el fitxer ex1.sol

Sortida per pantalla del resultat de l'execució si s'ha realitzat correctament

Després d'executar el programa glpsol, si s'ha pogut interpretar el model, a la terminal hi apareix el missatge següent:

```
GLPSOL: GLPK LP/MIP Solver, v4.45
Parameter(s) specified in the command line:
 -m ex1.mod -o ex1.sol
Reading model section from ex1.mod...
17 lines were read
Generating z...
Generating r1...
Generating r2...
Generating r3...
Model has been successfully generated
GLPK Simplex Optimizer, v4.45
4 rows, 2 columns, 7 non-zeros
Preprocessing...
2 rows, 2 columns, 4 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  3.000e+00  ratio =  3.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part = 2
*     0: obj =   0.000000000e+00  infeas =  0.000e+00 (0)
*     3: obj =   1.625000000e+01  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (114817 bytes)
Writing basic solution to 'ex1.sol'...
```

L'informe mostra que el glpsol llegeix el programa, crida una funció GLPK per generar la funció objectiu després una altra per generar les restriccions. Quan està llegit correctament el model, explica breument com s'executa internament i al final ens dóna informació sobre si hi ha solució.

Sortida per pantalla quan el programa ha trobat errors en el codi

Un exemple podria ser que ens deixéssim un punt i coma al final de la línea 11. Apareix per pantalla

```
GLPSOL: GLPK LP/MIP Solver 4.38
Reading model section from ex1.mod...
ex1.mod:15: syntax error in constraint statement
Context:  + 2 * y <= 10 ; subject to r2 : 3 * x + 2 * y <= 15 subject
MathProg model processing error
```

Ens avisa que a la línea 12 hi ha un error. Si ens fixem en el missatge

```
r2 : 3 * x + 2 * y <= 15 subject
```

veiem que s'ha aturat quan ha trobat `subject to` a la línea 12 sense que tanquéssim la declaració de la restricció de la línea 11.

Execució de glpsol demanant l'escriptura de la solució en un nou fitxer. Lectura del resultat

Hem demanat a glpsol que escrivís la solució en el fitxer `ex1.sol`. L'obrim amb el mateix editor de fitxers. La solució està dividida en quatre seccions:

- Informació sobre el problema i del valor optimal de la funció ojectiu

  ```
  Problem:    ex1
  Rows:       4
  Columns:    2
  Non-zeros:  7
  Status:     OPTIMAL
  Objective:  z = 16.25 (MAXimum)
  ```

  Veiem que ha trobat la solució optimal, que la funció z assoleix el màxim $z = 16.25$.

- Informació precisa sobre els valors i estatus de la funció objectiu i de les restriccions

  ```
  No.   Row name   St   Activity     Lower bound Upper bound   Marginal
  --- ------------ -- ------------- ------------------------ ---------
    1 z            B          16.25
    2 r1           NU            10                       10       1.25
    3 r2           NU            15                       15       0.25
    4 r3           B           3.75                        4
  ```

  Al costat dels noms de la funció objectiu i de les restriccions apareixen l'estatus (St), l'activitat, si està acotada superiorment o inferiorment, i el valor marginal també anomenat sensitivitat.

  L'activitat és el valor de la funció objectiu i les funcions lineals de les restriccions en el punt optimal.

  L'estatus (St) en les restriccions, indica si s'ha assolit la cota o no. En aquest cas totes les restriccions imposaven una cota superior. Fixem-nos en l'activitat: les dues primeres arriben al topall, a 10 i 15 respectivament (s'anomenen **restriccions actives**). En canvi r3 no hi arriba, val 3.75, quan la restricció imposava cota superior 4. L'estatus `NU` (si fos cota inferior `NL`) indica que s'ha assolit la cota superior (o inferior). La lletra `B` indica que no s'ha assolit.

  La columna `Marginal` dóna, en els casos en que s'assoleix la cota (r1 i r2), un número, la seva **sensitivitat** o preu ombra. Aquestes dues cotes impedeixen que la funció objectiu creixi,

per això s'anomenen restriccions actives. Si les relaxéssim (en aquest cas les incrementéssim), la funció objectiu milloraria. El número `1.25` que hi ha per a la restricció r1 indica que si augmentéssim en δ unitats el valor de la cota superior que imposa aquesta restricció, la funció objectiu augmentaria en $1.25\,\delta$ unitats. Anàlogament, si augmentéssim en δ unitats la cota superior de la restricció r2, deixant la resta de restriccions igual, la funció objectiu augmentaria en $0.25\,\delta$. En canvi la funció objectiu no variaria si canviéssim una mica la cota superior de la restricció r3, perquè no s'assoleix (la restricció no és activa) i el punt optimal seguiria essent al mateix.

- A continuació el fitxer solució ens dóna el punt optimal, el valor de cada variable.

```
   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -----------
     1 x            B            2.5             0
     2 y            B           3.75             0
```

- Per acabar l'arxiu ex1.sol ens dóna informació més tècnica sobre les condicions d'optimalitat.

1.2 Variables indexades i sumatoris

Considerem el programa lineal:

$$\text{minimitzar} \quad z = x_1 + x_2 - 2x_3 + x_4$$

$$\text{subjecte a:} \quad \begin{cases} x_1 - x_2 - x_3 - 2x_4 \geq 2 \\ x_1 + x_2 \quad\quad - \;\; x_4 \leq 8 \\ x_1 + x_2 + x_3 + \;\; x_4 = 4 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Per resoldre'l, podem escriure el següent fitxer ex2.mod

```
# Exemple 2.
# Declaració de variables indexades.
var x{1..4}>=0;

# Funció objectiu
minimize z:  x[1]+x[2]-2*x[3]+x[4];

# Restriccions
subject to
r1:  x[1]-x[2]-x[3] -2*x[4]>= 2;
r2: x[1]+x[2]-x[4] <= 8;
r3: sum{i in 1..4} x[i]= 4;

end;
```

A banda que ara cal demanar `minimize` (minimitzar la funció objectiu), en aquest exemple hi ha dues diferències amb l'exemple anterior:

En primer lloc, les variables tenen subíndexs d'1 a 4. Per això hem modificat la declaració de les variables, que ara és una de sola: `var x{1..4}>=0;`

Fixem-nos que a l'hora de referir-nos dins les fórmules a una de les variables concreta, les hem denotat per `x[1]`, `x[2]`, `x[3]`, `x[4]`

En aquest exemple, a la restricció r3, hem expressat la suma $x_1 + x_2 + x_3 + x_4$ de forma curta, amb un sumatori: `sum{i in 1..4} x[i]`

1.3 Declaració de paràmetres, paràmetres vectors i bloc de dades

Una fàbrica produeix cable elèctric d'alta qualitat usant dos tipus d'aliatges metàlics, A i B. L'aliatge A conté un 80% de coure i un 20% d'alumini. L'aliatge B conté un 68% de coure i un 32% d'alumini. L'aiatge A té un preu de 8000 euros per tonelada i el B, 6000 euros per tonelada.

Quines quantitats de cada aliatge ha d'usar l'empresa si vol produir una tonelada de cable que contingui almenys un 23% d'alumini i almenys un 70% de coure per tal que el cost de producció sigui el mínim possible?

Per a trobar la solució del problema amb GLPK, escrivim el codi següent:

```
1  # declaració dels paràmetres:
2  param produccio;
3  param costAliatge{1..2};
4  param propCu{1..2};
5  param propAl{1..2};
6
7  var x{1..2}>=0; # quantitat de cada aliatge
8
9  minimize cost: sum{i in 1..2} costAliatge[i]*x[i];
10
11 subject to
12 totalprod: sum{i in 1..2} x[i]=produccio;
13 alumini: sum{i in 1..2} propAl[i]*x[i] >=0.23*produccio;
14 coure: sum{i in 1..2} propCu[i]*x[i] >=0.7*produccio;
15
16 # Inici del bloc de dades:
17 data;
18
19 # donem valors als paràmetres que hem declarat:
20 param produccio:= 1;
21 param costAliatge:= 1 8000 2 6000;
22 param propCu:= 1 0.8 2 0.68;
23 param propAl:= 1 0.2 2 0.32;
24
25 end;
```

En aquest exemple utilitzem per primer cop els paràmetres. Aquests s'utlilitzaran dins les fórmules de la funció objectiu o de les restriccions, per tant s'han de declarar abans. Al final del fitxer els hi assignarem un valor. Els paràmetres poden ser números (en aquest cas "produccio"), vectors (costAliatge, propCu i propAl) o matrius (com veurem en l'exemple següent).

- A la línia 2 declarem el paràmetre producció que contindrà la quantitat de cable que volem produir (només el declarem): `param produccio;`
- A la línia 3 declarem el cost de cada l'aliatge. Com que hi ha dos aliatges, que dins del programa porten els índexs 1 i 2, el cost de l'aliatge ha de ser un vector: `param costAliatge{1..2};`

- A la línia 4 declarem el paràmetre de proporció de coure de cada aliatge. Com que hi ha els aliatges 1 i 2, es tracta d'un vector indexat a 1 i 2: `param propCu{1..2};` El mateix fem amb el paràmetre `propAl` de proporció d'alumini.
- A la línia 9 veiem que ara podem escriure la funció objectiu en termes del paràmetre costAliatge de manera molt més senzilla, utlilitzant un sumatori tal com faríem en un paper: $\sum_{i=1}^{2} c_i\, x_i$, (on c_i seria el cost de l'aliatge i). En glpk s'escriu:
`sum{i in 1..2} costAliatge[i]*x[i];`
- A les línies 12–14 escrivim les restriccions en termes dels paràmetres `produccio`, `propAl` i `propCu`.
- A la línia 17 comença el bloc de dades amb l'ordre `data;`
- A la línia 20 donem valor al paràmetre producció.
- A les línies 21–23 donem valor als paràmetres–vector `costAliatge`, `propAl` i `propCu`. Fixem-nos en la sintàxi tan particular.

1.4 Utilització de conjunts d'índexs

En el problema anterior teníem dos aliatges, A i B, que en el codi de glpk hem anomenat 1 i 2. Podríem tenir la necessitat de deixar obert el nombre i el nom dels aliatges, per si en un futur treballem amb més varietats. El codi anterior es pot millorar afegint un conjunt d'aliatges, que s'ha de declarar al principi com `set aliatges;` i canviant després la declaració dels paràmetres–vector que tenien dues components (tantes components com aliatges). També cal canviar la declaració de la variable–vector. Com hem fet amb els paràmetres, en el bloc de dades del final del fitxer especifiquem els elements del conjunt aliatges.

```
1  #declaració d'un conjunt d'índexs
2  set aliatges;
3  param produccio;
4  param costAliatge{aliatges};
5  param propCu{aliatges};
6  param propAl{aliatges};
7
8  var x{aliatges}>=0;
9
10 minimize cost: sum{i in aliatges} costAliatge[i]*x[i];
11
12 subject to
13 totalprod: sum{i in aliatges} x[i]=produccio;
14 alumini: sum{i in aliatges} propAl[i]*x[i] >=.23*produccio;
15 coure: sum{i in aliatges} propCu[i]*x[i] >=.7*produccio;
16
17 data;
18 set aliatges:= A B;
19 param produccio:= 1;
20 param costAliatge:= A 8000 B 6000;
21 param propCu:= A 0.8 B 0.68;
22 param propAl:= A 0.2 B 0.32;
23
24 end;
```

La manera anterior de procedir és molt útil perquè permet fer petits canvis en les dades només si varia el conjunt d'aliatges, o els seus preus, o les seves especificacions de composició. També facilita la lectura de la solució en el fitxer de sortida.

1.5 Declaració de paràmetres–matrius i restriccions indexades.

$$\text{maximitzar} \quad 4x_1 + x_2 + 3x_3 + 2x_4$$

$$\text{subjecte a:} \quad \begin{cases} 20x_1 + 20x_2 + 10x_3 + 20x_4 \leq 6 \\ 20x_1 + 30x_2 + 30x_3 + 50x_4 \leq 10 \\ 40x_1 + 30x_2 + 10x_3 + 20x_4 \leq 11 \\ 20x_1 + 30x_2 + 10x_3 + 20x_4 \leq 7 \\ 20x_1 + 20x_2 \qquad\quad + 40x_4 \leq 8 \\ 20x_1 + 20x_2 + 20x_3 + 40x_4 \leq 12 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

```
1  # Declarem el paràmetre-vector dels termes independents de les restriccions
2  #                                (6 components)
3  param b{1..6};
4  # Declarem el paràmetre-vector dels coeficients
5  #         de la funció objectiu (4 components)
6  param c{1..4};
7
8  # Declarem la matriu de coeficients de les variables en les restriccions
9  #              un paràmetre-matriu   (6 files i 4 columnes)
10 param a{1..6,1..4};
11
12 var x{1..4}>=0;       # Variables (vector de 4 components)
13
14 # Declarem la funció ojectiu en termes dels paràmetres c:
15 maximize z : sum{i in 1..4} c[i]*x[i];
16
17 # Declarem les restriccions en termes dels paràmetres a i b:
18 subject to
19 r{i in 1..6}: sum{j in 1..4} a[i,j]*x[j] <= b[i];
20
21 # Seguidament afegim una secció de dades on introduim els valors dels
      paràmetres
22 data;
23
24 param a:
25      1    2    3    4 :=
26 1   20   20   10   20
27 2   20   30   30   50
28 3   40   30   10   20
29 4   20   30   10   20
30 5   20   20    0   40
31 6   20   20   20   40;
32
33 param b:=
34 1 6 2 10 3 11 4 7 5 8 6 12;
35
36 param c:=
37 1 4 2 1 3 3 4 2;
38 end;
```

Hem utilitzat per primer cop un paràmetre–matriu. L'hem declarat a la línia 10 posant ordenats els dos conjunts d'índexs, el de files que varia de 1 a 6 i el de columnes que ho fa de 1 a 4: `param a{1..6,1..4};`

Després, en les línies 24–31 dins del bloc de dades, hem donat valor a la matriu `a`, de manera molt específica, però bastant intuïtiva. S'escriu com una matriu llevat de l'encapçalament i l'inici de cada fila. Ben bé com una taula, amb capçaleres de files i de columnes. *En glpk podem deixar tants espais en blanc com vulguem, aquí ajuden a visualitzar la matriu.*

1.6 Exercicis

1. Digues quina és la solució optimal i quina és el valor òptim de la funció objectiu per al problema:

$$\begin{array}{ll} \text{minimitzar} & 18\,x_1 + 12\,x_2 + 2\,x_3 + 6\,x_4 \\ \text{subjecte a:} & \left\{\begin{array}{r} 3\,x_1 + x_2 - 2\,x_3 + x_4 = 2 \\ x_1 + 3\,x_2 - x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{array}\right. \end{array}$$

2. Digues quina és la solució optimal i quina és el valor òptim de la funció objectiu per al problema:

$$\begin{array}{ll} \text{maximitzar} & 4\,x_1 + x_2 + 3\,x_3 + 2\,x_4 \\ \text{subjecte a:} & \left\{\begin{array}{r} 2\,x_1 + 2\,x_2 + x_3 + 2\,x_4 \leq 6 \\ x_2 + 2\,x_3 + 3\,x_4 \leq 4 \\ 2\,x_1 + x_2 \leq 5 \\ x_2 \leq 1 \\ -x_3 + 2\,x_4 \leq 2 \\ x_3 + 2\,x_4 \leq 6 \\ x_1, x_2, x_3, x_4 \geq 0 \end{array}\right. \end{array}$$

3. Digues quina és la solució optimal i quina és el valor òptim de la funció objectiu per al problema:

$$\begin{array}{ll} \text{minimitzar} & 5\,x_1 - 6\,x_2 + 7\,x_3 + x_4 \\ \text{subjecte a:} & \left\{\begin{array}{r} x_1 + 2\,x_2 - x_3 - x_4 = -7 \\ 6\,x_1 - 3\,x_2 + x_3 + 7\,x_4 \geq 14 \\ -2\,x_1 - 7\,x_2 + 4\,x_3 + 2\,x_4 \leq -3 \\ x_1 + 3\,x_2 - x_4 = 2 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{array}\right. \end{array}$$

   I quan substituïm la restricció $x_2 \geq 0$ per $x_3 \geq 0$?

4. Un granger vol que cadascuna de les seves vaques rebi diàriament entre 16000 i 18000 calories, almenys 2 quilograms de proteïnes, i com a mínim 3 grams de vitamines. Disposa de tres tipus d'aliments que pot barrejar en qualsevol proporció, amb les característiques que s'indiquen a continuació (per quilogram).

Aliment	Cost(euros)	Calories	Proteïnes(kg)	Vitamines(gr)
1	0.8	3600	0.25	0.7
2	0.6	2000	0.35	0.4
3	0.2	1600	0.15	0.25

   Quina quantitat de cada aliment li ha de donar a cada vaca el granger si vol fer mínim el cost de manteniment de les vaques sense violar les condicions anteriors.

5. En un peatge d'autopista el nombre mínim de cobradors segons la franja horària és

Franges horàries	2-6	6-10	10-14	14-18	18-22	22-2
Cobradors necessaris	4	8	10	7	12	4

   (a) Si es segueix un horari normal de tres torns (matí de 8 a 16, tarda de 16 a 24, i nit de 0 a 8), a quina quantitat mínima de persones caldrà contractar?

   (b) En comptes d'aquests torns, considerem 6 possibles torns. Així, cada cobrador comença a treballar al principi d'un dels blocs de 4 hores de la taula i s'hi està vuit hores, en dos torns consecutius de 4 hores. Quina és la quantitat mínima de persones que caldrà contractar?

   (c) Estudieu aquests dos problemes si la quantitat mínima de cobradors durant els sis períodes successius fos

      i. 4, 12, 10, 7, 12, 4.

      ii. 4, 8, 7, 7, 12, 4.