SET PACKING:

M -> Puntos de demanda (lo que hay que cubrir)

N -> Puntos de instalación para cubrir la demanda según condiciones

EJEMPLO PUBLICIDAD CASO 2 PAG6

declarations

n = 5 ! numero de periodicos = puntos de servicio

pservicio = 1..n

m = 9 ! numero de regiones a donde la publicidad del periodico puede llegar

pdemanda = 1..m

a:array(pservicio,pdemanda)of integer

costo, utilidad: array(pservicio)of integer

x:array(pservicio)of mpvar

end-declarations

!Funcion objetivo

utilidad\_total:=sum(j in pservicio)utilidad(j)\*x(j)

!Restricciones cada ! regi�n como mucho tendr� un peri�dico

forall(i in pdemanda) r(i):=sum(j in pservicio)a(j,i)\*x(j)<=1

forall(j in pservicio) x(j) is\_binary

!Encontrar optimo y escribir solucion

maximize(utilidad\_total)

forall(j in pservicio)do

if(x(j).sol>0.9999)then

writeln("En periodico ",j," ponemos publicidad")

end-if

end-do

forall(i in pdemanda,j in pservicio)do

if(a(j,i)=1 and x(j).sol>0.9999)then

writeln("Region ",i," llega anuncio de periodico ",j)

else

writeln("Region ",i," NO llega anuncio de periodico ",j)

end-if

end-do

EJEMPLO SCP EJ2 4 AMBULANCIA

(declarations y lectura datos del fichero opciones\_lectura: opción 3)

! Modelo:

costo\_total:=sum(j in pservicio)costo(j)\*x(j)

forall (i in pdemanda)

res(i):=sum(j in pservicio) a(i,j)\*x(j)>=1 !Garantiza que hay minimo 1 ambulancia para cada region-> setcovering

forall (j in pservicio) x(j) is\_binary

minimize(costo\_total)

!Escribir solucion

writeln("\n")

writeln("El coste m�nimo es: ", getobjval)

!Localizaciones de las ambulancias

forall(j in pservicio) do

if (x(j).sol>0.999) then

writeln ("Se instala la ambulancia en localizacion ", j)

end-if

end-do

!Qu� localizacion de ambulancia da servicio a cada municipio

forall (i in pdemanda, j in pservicio) do

if (a(i,j)=1 and x(j).sol>0.9999) then

writeln("El municipio ", i , " es atendido por ambulancia localizada en ", j)

end-if

end-do

EJEMPLO SCP EJ2 5 CUBRIMIENTO GRAFOS

(declarations y lectura datos del fichero opciones\_lectura: opción 2)

forall(j in pservicio)costo(j):=1

writeln("Problema Set Covering, m = ",m,", n = ",n)

writeln("Datos: ",archivo\_datos)

writeln("dc = ",dc)

! Modelo:

costo\_total:=sum(j in pservicio)costo(j)\*x(j)

forall(i in pdemanda)rescub(i):=sum(j in pservicio)a(i,j)\*x(j)>=1

forall(j in pservicio)x(j)is\_binary

exportprob(EP\_MIN,"",costo\_total)

minimize(costo\_total)

writeln("Costo minimo: ",getobjval)

writeln("Puntos abiertos: ")

forall(j in pservicio|x(j).sol>=0.99)write(j,"\t")

writeln("")

!Salidas

forall (i in pdemanda, j in pservicio) do

if (a(i,j)=1 and x(j).sol>0.9999) then

writeln("El municipio ", i , " es atendido por ambulancia localizada en ", j)

end-if

end-do

**EXAMEN 2**

EJEMPLO SCP DC FIJO

! Calcular la matriz de distancias Euclideas:

forall(i,j in puntos)dist(i,j):=sqrt((cx(i)-cx(j))^2+(cy(i)-cy(j))^2)

! Modelo Set Covering con distancias

forall(j in puntos)x(j)is\_binary

! Funcion objetivo: numero de instalaciones

numin:=sum(j in puntos)x(j)

! Restricciones de cubrimiento: (se puede construir la matriz a(i,j) pero aqui la forma sin hacerlo->)

forall(i in puntos) rcub(i):=sum(j in puntos|dist(i,j)<=dc)x(j)>=1

!Resolver el modelo

minimize(numin)

!Salidas

writeln("dc = ",dc,", Numero de instalaciones: ",getobjval)

writeln("Instalaciones abiertas:")

forall(j in puntos|x(j).sol>=0.999)write(j," ")

EJEMPLO SCP VARIABLE

! Calcular la matriz de distancias Euclideas:

forall(i,j in puntos)dist(i,j):=sqrt((cx(i)-cx(j))^2+(cy(i)-cy(j))^2)

forall(j in puntos)x(j)is\_binary

numin:=sum(j in puntos)x(j)

forall(k in 200..500)do !cambiamos las diferentes posibilidades de distancia cubrimiento

dc:=k

forall(i in puntos) rcub(i):=sum(j in puntos|dist(i,j)<=dc)x(j)>=1

minimize(numin)

!writeln("dc = ",dc,", Numero de instaciones: ",getobjval)

writeln(dc,"\t",getobjval)

IVEdrawpoint(id1,dc,getobjval)

end-do

writeln("Instalaciones abiertas:")

forall(j in puntos|x(j).sol>=0.999)write(j," ")

EJEMPLO CREW SCHEDULING

(rutas son puntos de servicio. Los vuelos son los puntos de demanda)

(formato datos=3)

! Modelo:

costo\_total:=sum(j in pservicio)costo(j)\*x(j)

forall (i in pdemanda)

res(i):=sum(j in pservicio) a(i,j)\*x(j)=1 !Garantiza que hay minimo 1 vuelo !yo puse >=, pero el profe tiene = . sale el mismo resultado

forall (j in pservicio) x(j) is\_binary

minimize(costo\_total)

!Escribir solucion

writeln("\n")

writeln("Coste minimo es: ", getobjval)

!Localizaciones

forall(j in pservicio) do

if (x(j).sol>0.999) then

writeln ("Vuelo al destino ", j)

end-if

end-do

forall (i in pdemanda, j in pservicio) do

if (a(i,j)=1 and x(j).sol>0.9999) then

writeln("El vuelo ", j , " atiende a la ciudad ", i)

end-if

end-do

EJEMPLO SCP HEURISTICA GREEDY ALEATORIZADO

(coger variables de greedy y greedy aleatorizado del fichero procedure\_intercambio\_aletario\_Set\_Covering funciones de examen)

(NECISITAMOS scp\_metodo\_greedy\_aleatorizado Y scp\_eliminacion\_columnas\_redundantes QUE ESTAN EN FUNCIONES DE EXAMENES)

writeln("\n\n(2) Metodo greedy aleatorizado N = ",N,", K = ",K)

scp\_metodo\_greedy\_aleatorizado

writeln("\nmejor z = ",zmejor)

writeln("Numero de puntos abiertos: ",sum(j in pservicio)xsolmejor(j))

EJEMPLO SCP FORMATOS HEURISTICA MEJORAESTOCASTICA

(coger variables de greedy del fichero procedure\_intercambio\_aletario\_Set\_Covering funciones de examen)

procedure scp\_metodo\_greedy

zheur:=0

mcub:=0

forall(j in pservicio)xsol(j):=0

forall(j in pservicio)fijado(j):=0

forall(i in pdemanda)cubierto(i):=0

while(mcub < m)do

! calcular primero d(j)

forall(j in pservicio|fijado(j)=0)do

d(j):=0

forall(i in pdemanda|cubierto(i)=0)do

if(a(i,j)=1)then

d(j):=d(j)+1

end-if

end-do

if(d(j)=0)then

fijado(j):=1

xsol(j):=0

end-if

end-do

! encontrar el minimo de costo(j)/d(j)

gmin:=999.

forall(j in pservicio|fijado(j)=0)do

if(costo(j)/d(j)<gmin)then

gmin:=costo(j)/d(j)

jmin:=j

end-if

end-do

! nuevo punto de servicio abierto

xsol(jmin):=1

fijado(jmin):=1

zheur:=zheur+costo(jmin)

forall(i in pdemanda|a(i,jmin)=1 and cubierto(i)=0)do

cubierto(i):=1

mcub:=mcub+1

end-do

end-do

end-procedure

(coger variables de mejora estocastica del fichero procedure\_intercambio\_aletario\_Set\_Covering funciones de examen)

procedure scp\_mejora\_estocastica

! solucion inicial

scp\_metodo\_greedy

scp\_eliminacion\_pservicio\_redundantes

writeln("Solucion greedy inicial z = ",zheur)

zmejor:=zheur

xmejor:=xsol

final:=0

iter:=0

nfracasos:=0

while(final=0)do

iter:=iter+1

! 1. parto de la solucion actual dada xsol, zheur

! 2. Elijo al azar una solucion en el entorno de la actual y hago el intercambio

scp\_intercambio\_aleatorio

! 3. examino la la nueva solucion

if(zheur1 < zheur)then ! hago el intercambio

xsol:=xsol1

zheur:=zheur1

nfracasos:=0

!writeln("\nIteracion ",iter,", zheur = ",zheur)

if(zheur < zmejor)then ! guardo la mejor solucion de entre las iteraciones que realizao

zmejor:=zheur

xmejor:=xsol

writeln("\nIteracion ",iter,", zheur = ",zheur)

end-if

else

nfracasos:=nfracasos+1

end-if

if(nfracasos = iter\_max\_sin\_mejora)then

final:=1

end-if

end-do ! final de while(final=0)

end-procedure

EJEMPLO SCP FORMATOS HEURISTICA THREALSHOLD ACCEPTING

(coger variables de threshold Accepting del fichero procedure\_intercambio\_aletario\_Set\_Covering funciones de examen)

procedure scp\_threshold\_accepting

writeln("Heuristica Theshold Accepting")

writeln("umbral\_ini = ",umbral\_ini,", alpha = ",alpha,", nfijo = ",nfijo)

! solucion inicial

scp\_metodo\_greedy

scp\_eliminacion\_pservicio\_redundantes

writeln("Solucion greedy inicial z = ",zheur)

zmejor:=zheur

xmejor:=xsol

iter:=0

umbral:=umbral\_ini

cont:=0

final:=0

t1:=gettime

while(final=0)do

iter:=iter+1

cont:=cont+1

!1. Calculo el costo de la solucion actual xsol

zheur:=sum(j in pservicio)costo(j)\*xsol(j)

!2. Elijo al azar una solucion en el entorno de la actual y hago el intercambio

scp\_intercambio\_aleatorio

!examino la nueva solucion

if(zheur1<=zheur)then

xsol:=xsol1

if(zheur1<zmejor)then

zmejor:=zheur1

xmejor:=xsol1

writeln("\nIteracion de mejora ", iter, ", zheur = ",zheur1, ", umbral = ", umbral)

cont:=0

end-if

else

if(cont<=nfijo)then

if(zheur1<=zheur+umbral)then !cambio a la solucion generada, aunque sea peor

xsol:=xsol1

end-if

else !Actualizo umbral

umbral:=umbral\*alpha

cont:=4

end-if

end-if

if(iter>iter\_max or umbral<umbral\_fin)then !Contabilizo numero de iteraciones o si estamos por debajo del umbral\_fin

final:=1

end-if

end-do

writeln("\nSolucion final: \nZmejor = ",strfmt(zmejor,10,2), ", iter = ", iter, ", umbral = ", umbral)

end-procedure

EJEMPLO SCP FORMATOS HEURISTICA SIMULATED ANNEALING

(coger variables de Simulated Annealing del fichero procedure\_intercambio\_aletario\_Set\_Covering funciones de examen)

procedure scp\_simulated\_annealing

writeln("\n(4) Heuristica Simulated Annealing")

writeln("temp\_ini = ",temp\_ini,", alpha = ",alpha,", nfijo = ",nfijo)

! solucion inicial

scp\_metodo\_greedy

scp\_eliminacion\_pservicio\_redundantes

writeln("Solucion greedy inicial z = ",zheur)

zmejor:=zheur

xmejor:=xsol

iter:=0

temp:=temp\_ini

cont:=0

final:=0

t1:=gettime

while(final=0)do

iter:=iter+1

cont:=cont+1

! 1. calculo el costo de la soluci?n actual xsol

zheur:=sum(j in pservicio)costo(j)\*xsol(j)

! 2. Elijo al azar una soluci?n en el entorno de la actual y hago el intercambio

scp\_intercambio\_aleatorio

! examino la la nueva soluci?n

if(zheur1<=zheur)then

zheur:=zheur1

xmejor:=xsol1

writeln("\nIteracion de mejora ",iter,", zheur = ",zheur1,", temp = ",temp, ", zmejor = ", zheur)

cont:=0

else

if(cont<=nfijo)then

ran:=random

pn:=exp(-(zheur1-zheur)/temp)

if(ran<=pn)then ! cambio a la soluci?n generada, aunque sea peor

xsol:=xsol1

end-if

else

temp:=alpha\*temp

cont:=0

end-if

end-if

if(iter > iter\_max or temp < temp\_fin)then

final:=1

end-if

end-do ! final de while(final=0)

writeln("\nSolucion final: \nzmejor = ",strfmt(zmejor,10,2),", iter = ",iter,", temp = ",strfmt(temp,8,5),", tiempo = ",strfmt(gettime-t1,5,2))

!)

end-procedure

**CUBRIMIENTO MAXIMO**

EJEMPLO 3\_2 PAG16 T2

declarations

n:integer

archivo\_datos = "ambulancias\_3-2.dat"

end-declarations

fopen(archivo\_datos,F\_INPUT)

readln(n)

declarations

puntos = 1..n

dist:array(puntos,puntos)of integer

dem:array(puntos)of integer

dc :integer

distrito:array(puntos)of integer

p:integer ! numero de ambulancias(instalaciones)

x,y:array(puntos)of mpvar

end-declarations

forall(i in puntos)do

read(distrito(i))

forall(j in puntos)do

read(dist(i,j))

end-do

read(dem(i))

end-do

fclose(F\_INPUT)

forall(i in puntos)do

writeln

forall(j in puntos) write (dist(i,j),"\t")

end-do

dc:= 2

writeln("Problema de Cubrimiento Maximo, n = ",n," distritos")

writeln("Solucion exacta")

forall(j in puntos)x(j)is\_binary ! variable de localizaci�n

forall(i in puntos)y(i)is\_binary ! variable de cubrimiento

poblacion\_cubierta:=sum(i in puntos)dem(i)\*y(i)

forall(i in puntos)rescub(i):=

sum(j in puntos|dist(i,j)<=dc)x(j)>=y(i) !sin la matriz a en este caso

forall(k in 1..n)do

p:=k !puedo instalar dessde una ambulancia hasta ambulancia en todos los puntos

pinst:= sum(j in puntos)x(j)=p

maximize(poblacion\_cubierta)

percent:=100\*poblacion\_cubierta.sol/sum(i in puntos)dem(i)

writeln("\n\*\*p = ",p," -> Poblacion cubierta = ",getobjval,", % ",percent)

writeln("\np\tpobcub\t %\n")

writeln(p,"\t",getobjval,"\t",percent)

write("\nPuntos abiertos: ")

forall(j in puntos|x(j).sol>=0.999)write(j,"\t")

write("\nPuntos cubiertos: ")

forall(i in puntos|y(i).sol>=0.99)write(i,"\t")

end-do

EJEMPLO LECDAT\_MCP1 P-FIJO

forall (i in pdemanda, j in pservicio)read(a(i,j))

fclose(F\_INPUT)

!Modelo de cubrimiento maximo

forall(i in pdemanda)dem(i):=1!demanda unitaria

!!!!!!Problema: Poner un anuncio en L18 p=5 periodicos para maximizar el numero de lectores del anuncia

!Encontrar ka solucion en el caso de que yo tenga dinero para poner 5 anuncios

poblacion\_cubierta:=sum(i in pdemanda)dem(i)\*z(i)

forall(i in pdemanda)rescub(i):=sum(j in pservicio)a(i,j)\*x(j)>=z(i)

rt:=sum(j in pservicio)x(j)=p

forall(j in pservicio)x(j)is\_binary

forall(i in pdemanda)z(i)is\_binary

!Resolvemos

maximize(poblacion\_cubierta)

writeln("p = ",p," numero de lectores: ",getobjval)

!Periodicos con publicidad

forall(j in pservicio)do

if(x(j).sol>0.999)then

writeln("El periodico ", j, " tiene publicidad")

end-if

end-do

!Lectores que ven el periodocio

forall(i in pdemanda)do

if(z(i).sol>0.999)then

writeln("El lector ",i, " ve el anuncio")

end-if

end-do

EJEMPLO LECDAT\_MCP1 P-VARIABLE

forall (i in pdemanda, j in pservicio)read(a(i,j))

fclose(F\_INPUT)

p:=5 !p inicial antes de p variable

writeln("problema de publicidad con p fijo, m=", m, "lectores, n=", n, "periodicos", p)

forall(j in pservicio)x(j)is\_binary ! variable de localizaci?n

forall(i in pdemanda)z(i)is\_binary ! variable de cubrimiento

!Modelo de cubrimiento maximo

forall(i in pdemanda)dem(i):=1!demanda unitaria

pinst:= sum(j in pservicio)x(j)=p

poblacion\_cubierta:=sum(i in pdemanda)dem(i)\*z(i)

forall(i in pdemanda)rescub(i):=

sum(j in pservicio|a(i,j)=1)x(j)>=z(i)

maximize(poblacion\_cubierta)

percent:=100\*poblacion\_cubierta.sol/sum(i in pdemanda)dem(i)

writeln("\n\*\*p = ",p," -> Poblacion cubierta = ",getobjval,", % ",percent)

writeln("\np\tpobcub\t %\n")

writeln(p,"\t",getobjval,"\t",percent)

write("\nPuntos abiertos: ")

forall(j in pservicio|x(j).sol>=0.999)write(j,"\t")

write("\nPuntos cubiertos: ")

forall(i in pdemanda|z(i).sol>=0.99)write(i,"\t")

forall(k in 1..n)do

p:=k

pinst:= sum(j in pservicio)x(j)=p

maximize(poblacion\_cubierta)

percent:=100\*poblacion\_cubierta.sol/sum(i in pdemanda)dem(i)

writeln("\n\*\*p = ",p," -> Poblacion cubierta = ",getobjval,", % ",percent)

writeln("\np\tpobcub\t %\n")

writeln(p,"\t",getobjval,"\t",percent)

write("\nPuntos abiertos: ")

forall(j in pservicio|x(j).sol>=0.999)write(j,"\t")

write("\nPuntos cubiertos: ")

forall(i in pdemanda|z(i).sol>=0.99)write(i,"\t")

end-do

end-model

EJEMPLO LECDAT\_MCP1 P-FIJO GREEDY

(coger variables de greedy del lecdat\_demanda\_tipo\_pln funciones de examen)

demandacubierta:=0

!METODO GREEDY

forall(t in 1..p)do !hacemos p iteraciones

!calculo Z(j) demanda aun no cubierta que cubre cada punto de servicio j

forall(j in pservicio|xsol(j)=0)do

Z(j):=0

forall(i in pdemanda)do

if (cubierto(i)=0 and a(i,j)=1) then

Z(j):=Z(j)+dem(i)

end-if

end-do

end-do

!Encuentro el maximo de Z(j)

zmax:=-999 !empiezo en numero negativo porque voy a maximizar

forall(j in pservicio|xsol(j)=0)do

if(Z(j)>zmax)then

zmax:=Z(j)

jmax:=j

end-if

end-do

xsol(jmax):=1

forall(i in pdemanda)do

if(cubierto(i)=0 and a(i, jmax)=1)then

cubierto(i):=1

demandacubierta:=demandacubierta+dem(i)

end-if

end-do

end-do

writeln("\nSolucion greedy(a cuantos puntos de demanda llega periodico):", demandacubierta)

forall(j in pservicio)writeln(" x(",j,") = ",xsol(j))

EJEMPLO LECDAT\_MCP1 P-VARIABLE GREEDY

(coger variables de greedy del lecdat\_demanda\_tipo\_pln funciones de examen)

! Metodo greedy

writeln("\nSolucion greedy:")

forall(k in 1..n)do

p:=k

metodo\_greedy\_mclp(p)

writeln("\np = ",p,"-> demanda cubierta ",demandacubierta)

end-do

!==============================================================

procedure metodo\_greedy\_mclp(p:integer)

forall(i in pdemanda)cubierto(i):=0

forall(j in pservicio)xsol(j):=0

demandacubierta:=0

forall(t in 1..p)do

! calculo Z(j) = demanda aun no cubierta que cubre cada j

forall(j in pservicio|xsol(j)=0)do

Z(j):=0

forall(i in pdemanda)do

if(cubierto(i)=0 and a(i,j)=1)then

Z(j):=Z(j)+dem(i)

end-if

end-do

end-do

! encuentro el maximo de Z(j)

zmax:=-999

forall(j in pservicio|xsol(j)=0)do

if(Z(j)>zmax)then

zmax:=Z(j)

jmax:=j

end-if

end-do

xsol(jmax):=1

forall(i in pdemanda)do

if(cubierto(i)=0 and a(i,jmax)=1)then

cubierto(i):=1

demandacubierta:=demandacubierta+dem(i)

end-if

end-do

end-do

end-procedure

EJEMPLO LECDAT\_MCP1 P-FIJO GREEDY+BUSQUEDA LOCAL (REVISAR RESTRICCIONES Y VARIABLES)

(coger variables de greedy del lecdat\_demanda\_tipo\_pln funciones de examen y función demanda\_que\_cubre)

1º.- HACEMOS GREEDY

!Busqueda local

busqueda\_local\_2

demandacubierta:=demanda\_que\_cubre(xsol)

writeln("\nSolucion busqueda local(a cuantos puntos de demanda llega periodico):", demandacubierta)

forall(j in pservicio)writeln(" x(",j,") = ",xsol(j))

procedure busqueda\_local\_2

final:=0

iter:=0

while(final = 0)do

iter:=iter+1

zheur:=demanda\_que\_cubre(xsol)

mejoramax:=-999

forall(k,s in pservicio|xsol(k)=1 and xsol(s)=0)do

xsolp:=xsol

xsolp(k):=0

xsolp(s):=1

zheurp:=demanda\_que\_cubre(xsolp)

mejora:=zheurp-zheur

if(mejora > mejoramax)then

mejoramax:=mejora

kmax:=k

smax:=s

end-if

end-do

if(mejoramax <=0)then

final:=1

else

xsol(kmax):=0

xsol(smax):=1

zheur:=zheur+mejoramax

end-if

end-do

**P-MEDIANA**

PRÁCTICA1\_PMEDIANA -> (Función de examen dibujar solución pmediana)

EJEMPLO 4-3 P-20

declarations

m = 4

n = 4

p = 2 !2 almacenes

pdemanda = 1..m

pservicio = 1..n

dem:array(pdemanda)of real

dist:array(pdemanda,pservicio)of real

x:array(pservicio)of mpvar

y:array(pdemanda,pservicio)of mpvar

end-declarations

! Datos del ejemplo:

dem::[3000, 2000, 2500, 2700]

dist::[

0, 25, 35, 40,

25, 0, 20, 40,

35, 20, 0, 30,

40, 40, 30, 0]

! Modelo:

dist\_total:=sum(i in pdemanda,j in pservicio)dem(i)\*dist(i,j)\*y(i,j) !y(i,j)=1 punto de demanda i va a ser atendido por el punto de servicio j

forall(i in pdemanda)r1(i):=

sum(j in pservicio)y(i,j)=1

r2:= sum(j in pservicio)x(j)=p

! forma desagregada: -> PERO TRABAJAREMOS CON LA AGREGADA SIEMPRE

forall(i in pdemanda,j in pservicio)r3(i,j):=

y(i,j)<=x(j)

! forma agregada:

forall(j in pservicio)r4(j):=

sum(i in pdemanda)y(i,j)<=m\*x(j)

forall(i in pdemanda, j in pservicio)y(i,j)is\_binary

forall( j in pservicio)x(j)is\_binary

exportprob(EP\_MIN,"",dist\_total)

!Resolucion:

minimize(dist\_total)

writeln("Distancia total: ",getobjval)

writeln("Puntos de servicio abierto:")

forall(j in pservicio|x(j).sol>=0.99)write(j,"\t")

writeln

forall(i in pdemanda,j in pservicio)

if(y(i,j).sol>=0.99)then ! a q punto de servicio se le apunta cada punto de demanda

writeln(i," -> ",j)

end-if

EJEMPLO 4-3 P-20 CON **GREEDY**

! variables para el método greedy:

Z:array(pservicio)of integer

dact:array(pdemanda)of integer

xsol:array(pservicio)of integer

ysol:array(pdemanda,pservicio)of integer

asign:array(pdemanda)of integer

BIGM = 99999999

daux:integer

jmin:integer

!==================== ALGORITMO GREEDY ===============================

!Inicializaciones:

forall(j in pservicio)xsol(j):=0

forall(i in pdemanda)dact(i):=BIGM

!t1:=gettime

forall(k in 1..p)do

!Etapa 1: se define el criterio para los puntos de servicio j que a?n no est?n en la soluci?n:

forall(j in pservicio|xsol(j)=0)do

Z(j):=0

forall(i in pdemanda)do

if(dist(i,j)<dact(i))then

Z(j):=Z(j)+dist(i,j)\*dem(i)

else

Z(j):=Z(j)+dact(i)\*dem(i)

end-if

end-do

end-do

! Etapa 2: se minimiza el criterio

daux:=BIGM

forall(j in pservicio)do

if(xsol (j)=0 and Z(j)<daux)then

daux:=Z(j)

jmin:=j

end-if

end-do

! Etapa 3: se introduce el m?nimo en la soluci?n

xsol(jmin):=1

ind\_ab(k):=jmin

! Etapa 4: se actualizan las distancias de los puntos de demanda a la soluci?n actual

forall(i in pdemanda)do

dact(i):=minlist(dact(i),dist(i,jmin)) ! o distancia que ya tenias o la del punto nuevo que acabas de abrir

end-do

end-do !Paramos cuando tengamos los p puntos de servicio abierto

z\_greedy:=sum(i in pdemanda)dact(i)\*dem(i) !puntos de servicio que se abren

! Soluci?n final

writeln("\n(2) Heuristica greedy\ndist total = ",z\_greedy)

!writeln("tiempo = ",strfmt(gettime-t1,5,2))

writeln("Puntos de servicio abierto en solucion greedy:")

forall(j in pservicio|xsol(j)>=0.99)write(j,"\t")

! De forma equivalente, que para eso se ha guardado el array ind\_ab (?ndic es de los puntos abiertos)

writeln

forall(j in 1..p)write(ind\_ab(j),"\t")

writeln

! Asignación de cada punto de demanda al punto de servicio abierto más cercano

! La asignación puede guardarse con un array de doble índice, como ysol(i,j), pero

! en realidad es más cómodo con un array simple asign(i) que el el punto

! de servicio abierto al que se asigna i.

forall(i in pdemanda)do

dmin:=BIGM

forall(j in pservicio|xsol(j)=1)do

if(dist(i,j)<dmin)then

dmin:=dist(i,j)

jmin:=j

end-if

end-do

asign(i):=jmin ! o con el array de doble índice

ysol(i,jmin):=1

writeln("punto de demanda ", i," -> ","punto de servicio ", jmin)

end-do

PMEDIANA-GRASP-BUSQUEDA-LOCAL-WHITAKER

(necesitamos variables greedy anteriores)

! variables para el metodo de intercambios de Teizt-Bart

ind\_ab:array(1..p)of integer

nuevo\_ind\_ab:array(1..p)of integer

ind\_noab, marcado, marca:array(pservicio)of integer

asignacio:array(pdemanda)of integer

nueva\_asignacio:array(pdemanda)of integer

temp, jd: integer

zheur,zmejor, mejora\_max, mejora:integer

bigM = 99999999

tolm = 1

!RCL (GRASP)

niter = 10

nlist = 5

!==================== ALGORITMO GRASP ===============================

writeln("\n(2) Metodo GRASP, niter = ",niter,", nlist = ",nlist)

zmejor:=BIGM

final:=0

t1:=gettime

forall(iter in 1..niter)do

!Inicializaciones:

forall(j in pservicio)xsol(j):=0

forall(i in pdemanda)dact(i):=BIGM

! Método de construcción greedy aleatorizado

forall(k in 1..p)do

!Etapa 1: se define el criterio para los j aún no en la solución:

forall(j in pservicio|xsol(j)=0)do

Z(j):=0

forall(i in pdemanda)do

if(dist(i,j)< dact(i))then Z(j):=Z(j)+dist(i,j)\*dem(i)

else Z(j):=Z(j)+dact(i)\*dem(i)

end-if

end-do

marca(j):=0 !puntos que entran en el RCL no cogidos con anterioridad

end-do

! Etapa 2: se encuentran los nlist indices con el mejor criterio

forall(f in 1..nlist)do

daux:=BIGM

forall(j in pservicio|xsol(j)=0 and marca(j)=0)do

if( Z(j)<daux)then

daux:=Z(j)

jmin:=j

end-if

end-do

marca(jmin):=1

ind(f):=jmin

end-do

! elijo al azar un elemento de la lista:

jmin:=ind(ceil(random\*nlist))

! Etapa 3: se introduce el mínimo en la solución

xsol(jmin):=1

!k in 1..p

ind\_ab(k):=jmin

! Etapa 4: se actualizan las distancias a la solución actual

forall(i in pdemanda)do

dact(i):=minlist(dact(i),dist(i,jmin))

end-do

end-do ! final de forall(k in 1..p)

z\_greedy:=sum(i in pdemanda)dact(i)\*dem(i)

write("\nIteracion ",iter," z = ",z\_greedy)

!Puntos no abiertos:

na:=0

forall(j in pservicio)do

if(xsol(j)=0)then

na:=na+1

ind\_noab(na):=j

end-if

end-do

if(na <>n-p) then writeln("\nAlgun error, na = ",na," no coincide con n-p = ",n-p);end-if

!============== METODO DE INTERCAMBIOS DE TEITZ-BART ================================

heur\_whitaker (en funciones de examen)

zheur:=calcular\_obj(ind\_ab) (en funciones de examen)

writeln(" solucion whitaker ", zheur)

if(zheur < zmejor)then zmejor:=zheur;end-if

end-do

PMEDIANA-SIMULATED ANNEALING

(necesitamos greedy y sus variables)

!============ Heurística Simulated Annealing =================================

writeln("\n(3) Heuristica Simulated Annealing")

writeln("temp\_ini = ",temp\_ini,", alpha = ",alpha,", nfijo = ",nfijo)

iter:=0

temp:=temp\_ini

cont:=0

final:=0

zheur:=z\_greedy

zmejor:=zheur

t1:=gettime

while(final=0)do

iter:=iter+1

cont:=cont+1

! 1. calculo el coste de la solución actual xsol

zheur:=calcular\_obj(xsol)

! 2. Elijo al azar una solución en el entorno de la actual y hago el intercambio

!============================================================================

intercambio\_aleatorio(xsol)

xsolp:=xsol

xsolp(jp):=0

xsolp(kp):=1

!======================================================================

! 3. examino la la nueva solución

zheur1:=calcular\_obj(xsolp)

if(zheur1<zheur)then

xsol:=xsolp

if(zheur1<zmejor)then

zmejor:=zheur1

xmejor:=xsolp

writeln("\nIteracion ",iter,", zheur = ",zheur1)

cont:=0

end-if

else

if(cont<=nfijo)then

ran:=random

pn:=exp(-(zheur-zheur1)/temp)

if(ran>=pn)then

xsol:=xsolp

end-if

else

temp:= alpha\*temp

cont:=0

end-if

end-if

if(iter > iter\_max or temp < temp\_fin)then

final:= 1

end-if

end-do

writeln("\nSolucion con simulated annealing: ",zmejor)

procedure intercambio\_aleatorio(xsol:array(range)of integer)

declarations

n1, n0:integer

end-declarations

n1:=0

n0:=0

forall(j in pservicio)do

if(xsol(j)=1)then

n1:=n1+1

ind1(n1):=j

elif(xsol(j)=0)then

n0:=n0+1

ind0(n0):=j

end-if

end-do

jp:=ind1(ceil(random\*n1))

kp:=ind0(ceil(random\*n0))

end-procedure

**P-CENTRO**

PCENTRO XPRESS MODELO OPTIMO

(variables en pcentro\_ayuda)

!======================= MODELO ================================

dist\_max:=w !funcion objetivo

forall(i in pdemanda)res1(i):= sum(j in pservicio)y(i,j)=1 ! cada cliente debe ser asignado a un solo centro de servicio.

forall(i in pdemanda)restmax(i):=sum(j in pservicio)dist(i,j)\*y(i,j)<=w ! la distancia entre el cliente i y el centro de servicio asignado no debe superar dist\_max.

res2:= sum(j in pservicio)x(j)=p !voy a poner p servicios

forall(j in pservicio)resvub2(j):= sum(i in pdemanda)y(i,j)<=m\*x(j) ! el número de clientes asignados a un centro de servicio no debe superar su capacidad m. // forall(i in pdemanda,j in pservicio)resvub1(i,j):= y(i,j)<=x(j) -> formulación fuerte

forall(j in pservicio)x(j)is\_binary

forall(i in pdemanda,j in pservicio)y(i,j)is\_binary

writeln("\nResolucion con Xpress\_optimizer, tiempo maximo: ",tiempo\_calculo)

setparam("XPRS\_MAXTIME",tiempo\_calculo)

minimize(dist\_max)

writeln("Distancia maxima: ",getobjval,", tiempo = ",gettime-time1)

PCENTRO TABLA COMPLETA (PCENTRO COMO PROBLEMA CUBRIMIENTO TOTAL)

(variables en pcentro\_ayuda el resto)

declarations

dc:integer

dcmin = 10 !Valor minimo de distancia que deberemos actualizar si no encontramos la slicion para el P deseado

dcmax = 50 !Valor maximo de distancia que deberemos actualizar si no encontramos la slicion para el P deseado

end-declarations

ninstal:=sum(j in pservicio)x(j)!Funcion objetivo del SC

forall(j in pservicio)x(j)is\_binary !Restriciones del SC

forall(k in dcmin..dcmax)do !Consideramos distintas distancias de cubrimiento enteras

dc:=k

forall(i in pdemanda)rescub(i):= sum(j in pservicio|dist(i,j)<=dc)x(j)>=1

minimize(ninstal)

writeln("\nTiempo de respuesta\t",dc,"\tTotal de instalaciones ",getobjval)

write("Instalaciones abiertas:")

forall(j in pservicio|x(j).sol>0.9999)do

write("\t",j)

end-do

end-do

PCENTRO BUSQUEDA DICOTOMICA

(variables en función\_calculo\_tiempo\_respuesta)

dem\_total:=sum(j in pdemanda)dem(j)

forall (j in pservicio) x(j) is\_binary

objetivo:=sum(j in pservicio) x(j)

! Inicializamos los par�metros del Algoritmo de B�squeda Dicot�mica

dcl:=0

dch:=(n-1)\*max(i in pdemanda,j in pservicio)dist(i,j)

iteracion:=1

final:=0

while (final=0) do

dc:=floor((dch+dcl)/2)

forall(i in pdemanda) res\_cubrimiento(i):=sum(j in pservicio | dist(i,j)<=dc)x(j)>=1 !todos los puntos de demanda tienes que tener punto de servicio

!Resolvemos el problema de Set Covering

minimize(objetivo)

p\_estrella:=getobjval

writeln(iteracion,"\t",dcl,"\t",dc,"\t",dch,"\t",p\_estrella)

if (p\_estrella<=p) then

dch:=dc

else

dcl:=dc+1

end-if

iteracion:=iteracion+1

if(dcl=dch)then

final:=1

! Ejecutamos el ultimo paso del bucle

dc:=dch !dch o dcl como son igulaes da igual

forall (i in pdemanda)res\_cubrimiento(i):=sum(j in pservicio|dist(i,j)<=dc)x(j)>=1

minimize(objetivo)

end-if

end-do

writeln(iteracion,"\t",dcl,"\t",dch,"\t",dc,"\t",getobjval)

writeln("Solucion final:")

writeln("\nTiempo de respuesta: ",dc)

writeln("El numero de instalaciones minimo para esta dc es: ",getobjval)

writeln("Tiempo: ",gettime-time1)

forall(j in pservicio)do

if(x(j).sol>=0.9999)then

writeln("Abierta instalaci�n ",j)

end-if

end-do

PCENTRO MEJORA ESTOCASTICA (Mostrado en clase)

(variables en heur\_greedy\_pcentro (greedy e intercambio 1-1 en fun))

heur\_greedy\_pcentro

writeln("\n(3) Procedimiento de mejora estocastica, iter\_max = ",iter\_max)

mejora\_estocastica\_pcentro

writeln("Valor final: ",zheur)

!===============================================================

procedure mejora\_estocastica\_pcentro

zmejor:=zheur

xmejor:=xsol

final1:=0

iter:=0

nfracasos:=0

while(final1=0)do

iter:=iter+1

!1. parto de la solucion actual dada con : xsol,zheur

!2. elijo al azar una solucion en el entorno de la actual y examino el intercambio

intercambio\_aleatorio\_pcentro

xsol1:=xsol

xsol(jp):=0

xsol(kp):=1

zp:=calcular\_tiempo\_respuesta(xsol1)

if(zp<zheur)then !hago el intercambio

xsol:=xsol1

zheur:=zp

nfracasos:=0

if(zheur<zmejor)then

zmejor:=zheur

xmejor:=xsol

writeln("\nIteracion ",iter,", zheur = ",zheur)

end-if

else

nfracasos:=nfracasos+1

end-if

if(nfracasos = iter\_max)then

final1:=1

end-if

end-do ! final de while(final=0)

end-procedure