**TEMA 5: PROGRAMACIÓN DE TAREAS**

1. IS : 1 MAQUINA SECUENCIACIÓN

!funcion objetivo

obj:=sum(j in tareas)x(j)

!restricciones

forall(j,k in tareas| j<>k and s(k)<f(j) and f(k)>s(j)) x(j)+x(k)<=1

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

1. IS: M MÁQUINAS

!funcion objetivo

obj:=sum(i in maquinas,j in tareas) z(i,j)

!restricciones

!cada maquina a lo sumo una tarea

forall(j in tareas) sum(i in maquinas) z(i,j)<=1

!solapamiento

forall(i in maquinas,j in tareas,k in tareas| j<>k and s(k)<f(j) and f(k)>s(j)) z(i,j)+z(i,k)<=1

forall(i in maquinas,j in tareas) z(i,j) is\_binary

1. IS : INTERVAL PARTIONING: MINIMO DE MAQUINAS

!funcion objetivo

obj:=sum(i in maq) y(i)

!restricciones

!todas las tareas deben ser asignadas

forall(j in tareas) sum(i in maq) w(i,j)=1

!si una maquinas no se usa,no puede tener tareas asignadas

forall(i in maq,j in tareas) w(i,j)<=y(i)

!solapamiento

forall(i in maq,j in tareas,k in tareas| j<>k and s(k)<f(j) and f(k)>s(j)) w(i,j)+w(i,k)<=1

forall(i in maq)y(i) is\_binary

forall(i in maq,j in tareas) w(i,j) is\_binary

TAREA 1: SE INTRODUCEN LAS PRIORIDADES

s, f,p: array(tareas) of integer -> METER EL P PARA LAS PRIORIDADES

forall(j in tareas)read(j,s(j),f(j),p(j)) -> LEER LAS PRIORDIDADES

!funcion objetivo

obj:=sum(j in tareas)x(j)\*p(j) -> añadir la prioridad en la función objetivo

!! salida pq el getobjetval es con prioridades

writeln("Problema de encontrar el máximo número de tareas con un sólo procesador")

writeln

suma:=0

forall(j in tareas| x(j).sol>0.99) suma:=suma+1

writeln("Se ejecutan ",suma," tareas")

write("Se ejecutan las tareas: ")

forall(j in tareas | x(j).sol>0.99) write(j," ")

writeln

sW:=0.

forall(j in tareas | x(j).sol>0.99) sW:=sW+p(j)

writeln

writeln("La prioridad maxima es: ", sW)

!y en el greedy considerar la prioridad

!Buscar el k que maximima la prioridad p(k)

kmin:=0

aux:=-999999

forall(j in tareas | marcada(j) = 1) do

if(p(j) > aux) then

aux:=p(j)

kmin:=j

end-if

end-do

Texto

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**PRÁCTICA 2. FORMULACIÓN DISYUNTIVA. (1||Ttotal\_w) -> tardanza total ponderada.**

!Defino la funcion objetivo: tardanza total ponderada

obj:=sum(j in tareas)(w(j)\*t(j))

!restricciones relativas a la fecha de entrega (linealizacion)

forall(j in tareas)res\_lin1(j):=t(j)>=0

forall(j in tareas)res\_lin2(j):=t(j)-s(j)=x(j)+p(j)-d(j)

forall(j in tareas)res\_lin3(j):= s(j)>=0

! Restricciones relativas al orden de los trabajos

forall(j,k in tareas |j<k) fd1(j,k):=x(j)+p(j)-x(k)<=M\*(1-y(j,k))

forall(j,k in tareas |j<k) fd2(j,k):=x(k)+p(k)-x(j)<=M\*y(j,k)

!2.3 no se incluye pq no tenemos release restriccion

forall(j in tareas)rt(j)<=x(j) !Restriccion de reléase -> seria esa

!restricciones de las variables

forall(j in tareas) x(j)>=0

forall(j,k in tareas |j<k) y(j,k) is\_binary

**TAREA 2. FORMULACIÓN DISYUNTIVA (1|prec|Lmax) -> 1 maquina, Precedencia, retraso máximo. (revisar en esta tarea y en la practica 2 como se meten los datos= ¡!!!!!!!!!!!!**

declarations

lmax:mpvar

end-declarations

!Defino la funcion objetivo

!minimizar el retraso maximo

obj:=lmax

!restricciones relativas a la fecha de entrega (linealizacion)

forall(j in tareas) lini(j):= lmax>=x(j)+p(j)-d(j)

! Restricciones relativas al orden de los trabajos

forall(j,k in tareas |j<k) fd1(j,k):=x(j)+p(j)-x(k)<=M\*(1-y(j,k))

forall(j,k in tareas |j<k) fd2(j,k):=x(k)+p(k)-x(j)<=M\*y(j,k)

!restricciones de precedencia

prec1:= x(1)+p(1)<= x(2) ! o bien y(1,2)=1

prec2:= x(2)+p(2)<= x(3) ! o bien y(2,3)=1

prec4:= x(4)+p(4)<= x(5) ! o bien y(4,5)=1

prec5:= x(4)+p(4)<= x(6) ! o bien y(4,6)=1

!restricciones de las variables

forall(j in tareas)x(j)>=0

forall(j,k in tareas|j<k)y(j,k) is\_binary

**PRÁCTICA 3. PROGRAMACIÓN DE TAREAS EN UNA MÁQUINA: [F1] ,[F2], HEURÍSTICAS Y METAHEURÍSTICAS. FUNCIÓN OBJETIVO TtotalW -> Tardanza Total Ponderada (PROBAR CON OTRAS FUNCIONES OBJETIVO Y VER EN LA TAREA 3 QUE HAY QUE TOCAR)**

PROBLEMA 1: F1 O FORMULACIÓN DISYUNTIVA -> DISCUTIDA EN LA PRÁCTICA Y TAREA 2

PROBLEMA 2: ESTA HECHA PERO HAY QUE CAMBIAR LA E SEGÚN EL PROBLEMA (están en transparencia 32 y 33 (revisar pq en la 32 no tiene e, pero es como tarea 2 .. linealización y tal) ->

forall(j in tareas) do

forall(t in 1..(T-p(j)+1)) do

e(j,t):=w(j)\*maxlist(0,t-1+p(j)-d(j)) !cambiar la e segun el problema de funcion objetivo

end-do

end-do

PROBLEMA 3: MULTISTART (PERMUTACIÓN + BÚSQUEDA LOCAL): HECHO PERO HAY QUE MODIFICAR calcularFuncionObjetivo según la función y búsqueda local : entender bien lo que hace y pensar posibles modificaciones.

PROBLEMA 4: GREEDY WSPT+BÚSQUEDA LOCAL: HECHO PERO HAY QUE MODIFICAR greedy\_wspt, búsqueda local y calcularFuncionObjetivo

PROBLEMA 5: GREEDY EDD+BÚSQUEDA LOCAL: HECHO PERO HAY QUE MODIFICAR greedy\_edd, búsqueda local y calcularFuncionObjetivo.

PROBLEMA 6: GREEDY WSPT + DESCENSO ESTOCÁSTICO: HECHO PERO HAY QUE MODIFICAR greedy\_wspt y calcularFuncionObjetivo. (revisar si hay que modificar descenso estocastico)

Sobre todo hay que tener en cuenta los intercambios en los métodos de mejora determinado con la función objetivo y como afectan a esta.

**TAREA 3. PROGRAMACIÓN DE TAREAS EN UNA MÁQUINA: [F1] ,[F2], HEURÍSTICAS Y METAHEURÍSTICAS. FUNCIÓN OBJETIVO CtotalW -> Tiempo de finalización total ponderado (REVISAR COMO SE INTRODUCEN LOS DATOS)**

* CAMBIAR LA COTA ACORDE A SI HAY RELEASE:

M := sum(j in tareas)p(j)+max(j in tareas)(rt(j))

* CAMBIAR LA T PARA LA F2 PARA TENER EN CUENTA EL RELEASE:

T := sum(j in tareas) p(j)+max(j in tareas)(rt(j))

* CAMBIAMOS LA E DE FORMULACIÓN GENERAL F2 (PAG 33):

forall(j in tareas) do

forall(t in 1..(T-p(j)+1)) do

e(j,t):=w(j)\*(t-1+p(j))

end-do

end-do

* GREEDY\_EDD\_RELEASE:

forall(t in tareas) criterio(t):=d(t)!criterio greedy

forall(t in tareas) marcado(t):=0!Ninguna tarea esta en la solucion

forall(t in tareas) orden(t):=0!No hay orden entre las tareas

sProc:=0

forall(i in 1..n)do

tmin:=0

cmin:=99999999999999.9

forall(t in tareas | marcado(t) = 0) do

if(criterio(t) < cmin and sProc>=rt(t)) then

tmin:=t

cmin:=criterio(t)

end-if

end-do

sProc:=sProc+p(tmin)

marcado(tmin):=1

orden(i):=tmin

end-do

* GREEDY\_WSPT\_RELEASE:

forall(t in tareas) criterio(t):=w(t)/p(t)

forall(t in tareas) marcado(t):=0

forall(t in tareas) orden(t):=0

sProc:=0

forall(i in 1..n)do

tmax:=0

cmax:=-999999999.

forall(t in tareas | marcado(t) = 0) do !ordenar decreciente w(t)/p(t) <-> ordenar creciente p(t)/w(t)

if(criterio(t) >= cmax and rt(t)<=sProc) then

tmax:=t

cmax:=criterio(t)

end-if

end-do

sProc:=sProc+p(tmax)

marcado(tmax):=1

orden(i):=tmax

end-do

* calcularFuncionObjetivo\_CTW -> para esto transparencias 20 y 21 . tiempofin= xj+pj

tiempofin:=0

forall(t in tareas)CT(t):=0.0

forall(t in tareas) do

tarea:=orden(t)

tiempofin:=tiempofin+p(tarea)

CT(tarea):=tiempofin

end-do

objetivo:=sum(j in tareas)w(j)\*CT(j)

returned:=objetivo

* BÚSQUEDA LOCAL: ATENCIÓN POR LOS INTERCAMBIOS

final:=0

while(final = 0) do

!Buscar la mejora m?xima

mejoramax:=0.0

f\_antes:=calcularFuncionObjetivo\_CTW(orden)

forall(tt1 in tareas, tt2 in tareas | tt1 < tt2) do

!TIEMPOS ACUMULADOS HASTA LA POSICIÓN ANTERIOR Y PONER EL NUEVO

tiempofin1:=0

forall(t in 1..tt1)do

if(t<>tt1)then

tarea:=orden(t)

tiempofin1:=tiempofin1+p(tarea)

end-if

if(t=tt1)then

tarea:=orden(tt2)

tiempofin1:=tiempofin1+p(tarea)

end-if

end-do

tiempofin2:=0

forall(t in 1..tt2)do

if(t<>tt2)then

tarea:=orden(t)

tiempofin2:=tiempofin2+p(tarea)

end-if

if(t=tt2)then

tarea:=orden(tt1)

tiempofin2:=tiempofin2+p(tarea)

end-if

end-do

!Probar el intercambio

temp:=orden(tt1)

orden(tt1):=orden(tt2)

orden(tt2):=temp

!tiempo release en ese intercambio

rel1:=rt(orden(tt1))

rel2:=rt(orden(tt2))

f\_despues:=calcularFuncionObjetivo\_CTW(orden)

!Deshacer el intercambio

temp:=orden(tt1)

orden(tt1):=orden(tt2)

orden(tt2):=temp

mejora:=f\_antes-f\_despues

if(mejora > mejoramax and (tiempofin1)<=(tiempofin2) and rel1<=(tiempofin1-p(orden(tt2))) and rel2<=(tiempofin2-p(orden(tt1)))) then

mejoramax:=mejora

t1max:=tt1

t2max:=tt2

end-if

end-do

if(mejoramax = 0) then

final := 1

else

!Se hace el intercambio

temp:=orden(t1max)

orden(t1max):=orden(t2max)

orden(t2max):=temp

end-if

end-do

**PRÁCTICA 4. PROGRAMACIÓN DE TAREAS DE MÁQUINAS EN PARALELO (P2|| Cmax -> tiempo máximo de finalización makespan) y (P2||TtotalW -> tardanza total ponderada) con [F2]**

**!**IMPORTANTE EN LAS SOLUCIONES REVISAR LAS INTERRUPCIONES PARA EL BALANCEO DE CARGA COMENTADO EN CLASE. COMENTAR EN EL EXAMEN QUE ES PARA ESO

!REVISAR EN EL EXAMEN LA LECTURA DE DATOS PORQUE PUEDE NO COINCIDIR EL FICHERO CON LA LECTURA QUE TENEMOS

FUNCIONES OBJETIVO Y RESTRICCIONES SABER: (PAG 57)

if(problema = 1) then

!funcion objetivo del problema 1

objetivo:=cmax

!restriccion propia del problema 1

forall(j in tareas) cmax>=c(j)

elif(problema = 2) then

!funcion objetivo del problema 2

objetivo:=sum(j in tareas) w(j)\*tardiness(j)

!restriccion propia del problema 2

forall(j in tareas) tardiness(j)>=c(j)-d(j)

forall(j in tareas) tardiness(j)>=0

end-if

! Después vienen las restricciones estándares. QUITAR LA DEL RELEASE SINO HAY

**TAREA 4. PROGRAMACIÓN DE TAREAS DE MÁQUINAS EN PARALELO (P3|| Cmax -> tiempo máximo de finalización makespan) MODELO EXACTO, GREEDY LPT, GREEDY LPT+heurística de intercambios 2-opt**

procedure greedy\_lpt

!1.-Ordenar de más a menos tiempos de procesado de las tareas

forall(j in tareas) orden(j):=0

qsort(SYS\_DOWN,p,orden) ! se almacena en orden las posiciones de p,ordenando decreciente

write(orden)

write(p)

!2.-Recorrer todas las tareas (bucle) e ir asinadolas a cada máquina

i1:=0

i2:=0

i3:=0

forall(i in 1..n) do

!2.1.-Seleccionamos la tarea actual

tareaActual:=orden(i)

!2.2.-Buscamos la máquina de menor carga con minlist y marcamos la maquina asignada

finMinimo:=minlist(finMaquina1,finMaquina2,finMaquina3)

write("Tarea ",tareaActual," asignada a ")

if(finMinimo = finMaquina1) then

maquinaAsignada:=1

writeln("máquina 1")

elif(finMinimo = finMaquina2) then

maquinaAsignada:=2

writeln("máquina 2")

else

maquinaAsignada:=3

writeln("máquina 3")

end-if

!2.3.-Se asigna el trabajo a la máquina correspondiente, asiganda en el paso anterior.

!Actualizar nº de tareas en la maquina, la posicion de la tarea en la maquina; tiempo de procesado de la maquina

if(maquinaAsignada = 1) then

i1:=i1+1

maquina1(i1):=tareaActual

finMaquina1:=finMaquina1 + p(tareaActual)

elif (maquinaAsignada = 2) then

i2:=i2+1

maquina2(i2):=tareaActual

finMaquina2:=finMaquina2 + p(tareaActual)

else

i3:=i3+1

maquina3(i3):=tareaActual

finMaquina3:=finMaquina3 + p(tareaActual)

end-if

end-do

end-procedure