UDPP (presa dalla tesi di Pacchiega, sezione 3.1.2)

d (i , j) = delay for flight fi in slot sj i in 1...n; j in 1...m

Decisional variables:

i in 1...n; j in 1…m

Objective function:

Tolto il termine f(d(i,j)) che rappresentava la funzione di costo di Cook, Tanner, Jovanovic, Lawes considerato che è praticamente lineare per ritardi inferiori ai 50 minuti

Constraints:

“The vector h represents in a system without protection the change of the non-prioritized flight delay”.

AMAL

d (i , j) = delay for flight fi in slot sj i in 1...n; j in 1...m

Decisional variables:

i in 1...n; j in 1…m

*d, f in 1…n*

Il passaggio dalla tipologia di offerta a 4 variabili a quella con 2 mantiene tutte le possibili soluzioni. L’offerta a 4 variabili permetterebbe agli utenti di sottoporre offerte più restrittive

Objective function:

Constraints:

Può essere eseguita al massimo un’offerta per volo.

Se nessuna offerta riguardante il volo *f* viene eseguita, il volo mantiene la posizione originale.

Implementato con .

Implementazione in Xpress del modello amal (non sono sicuro della forma della formulazione matematica, l’implementazione probabilmente non è né efficiente né elegante ma funziona):

forall(i in F,j in S) do

x(i,j) is\_binary

k(i,j) is\_binary

l1(i,j) is\_binary

l2(i,j) is\_binary

end-do

forall(fd in F,fu in F) do

o(fd,fu) is\_binary

end-do

forall(i in F) sum (j in F) o(i,j) + sum(j in F) o(j,i)<=1 !max 1 offer executed

forall(i in F,j in S) do

l1(i,j) <= x(i,j)

l1(i,j) <= 1-x\_FPFS(i,j)

l1(i,j) >= x(i,j) - x\_FPFS(i,j) !(x + (1-x\_F) -1)

l2(i,j) <= 1 - x(i,j)

l2(i,j) <= x\_FPFS(i,j)

l2(i,j) >= x\_FPFS(i,j) - x(i,j) !((1-x) + x\_F -1)

k(i,j) >= l1(i,j)

k(i,j) >= l2(i,j)

k(i,j) <= l1(i,j)+l2(i,j) !k(i,j) = x(i,j) XOR x\_FPFS(i,j)

k(i,j) <= sum (f in F)o(i,f) + sum (f in F)o(f,i) !slot change iff one offer for the slot is executed

end-do

forall(fd in F,fu in F) do

o(fd,fu) <= sum (j in 1..(fu-1)) x(fu,j)

o(fd,fu) <= sum (j in fd+1..NumberSlots) x(fd,j)

end-do

forall(i in F) sum (j in S) x(i,j) = 1

forall(j in S) sum (i in F) x(i,j) <= 1

forall(i in F,j in S) d(i,j):= Slot(j) - TInit(i)

forall(i in F,j in S) x(i,j)\*d(i,j)>=0 ! no negative delay

Costs := sum(i in F,j in S) (x(i,j)\*d(i,j)\*CMD(i))

minimize(Costs)