FICO Xpress Mosel | MILP | Hub Location | Profit Maximisation

Airline Hub Location Optimisation

Base_revised.mos — Annotated Source Code

FICO Xpress Mosel · MIS41160 Optimisation in Business · 2025

12-city US–France freight network · 3-hub MILP · Profit maximisation

This document presents the complete annotated source code for the **Base_revised.mos** Mosel model — a Mixed Integer Linear Programme that selects three optimal airline hub locations across a 12-city US–France freight network to maximise total profit under budget, environmental, and operational constraints.

How to Run

1. Place **Base_revised.mos** and **f4hubs.dat** in the same folder. 2. Open **FICO Xpress Mosel**. 3. Open the folder and select both files. 4. Run **Base_revised.mos**.

01 / MODEL HEADER & IMPORTS

Declares the model name and imports the Xpress solver library.

```
model "F-4 Hubs (expanded 12 cities - profit + environment)"
  uses "mmxprs"

  forward function calc_cost(i,j,k,l: integer): real
  forward function calc_cost_direct(i,j: integer): real
  forward function calc_revenue(i,j: integer): real
```

02 / INDEX SETS & NETWORK PARAMETERS

Defines the 12-city network split into US (1–6) and EU (7–12) airports. BASE airports are always open; NEW airports are candidate expansions subject to the budget constraint. NHUBS = 3 forces exactly three hubs.

```
declarations
  US   = 1..6     ! ATL, BOS, CHI, DFW, LAX, DEN
  EU   = 7..12    ! MRS, NCE, PAR, LIL, LYS, NTE
  CITIES = US + EU

  BASE = {1,2,3,7,8,9}    ! Always-open airports
  NEW  = {4,5,6,10,11,12} ! Candidate new airports

  NHUBS        = 3         ! Exactly 3 hubs
  BUDGET_OPEN  = 150000000 ! $150M opening budget
  MIN_NEW_OPEN = 2         ! At least 2 new airports

  REV_DOM      = 3.0       ! $/mile domestic
  REV_INT      = 4.0       ! $/mile international
  EMISS        = 250.0     ! Penalty per flight leg/unit
  ONE_HUB_FACTOR = 0.9     ! 10% discount for one-hub
```

03 / DECISION VARIABLES

Four binary variable families: open_i (airport active?), hub_i (airport a hub?), direct_ij (freight sent direct?), flow_ijkl (freight routed via hubs k and l?). Flow variables are created dynamically — only for feasible routing patterns.

```
  QUANT: array(CITIES,CITIES) of integer  ! Demand between cities
  DIST:  array(CITIES,CITIES) of integer  ! Distance in miles
  OPEN:  array(CITIES) of real            ! Hub opening cost
  NAMES: array(CITIES) of string          ! City labels
  FACTOR: real                            ! Inter-hub discount

  flow:   dynamic array(CITIES,CITIES,CITIES,CITIES) of mpvar
  direct: array(CITIES,CITIES) of mpvar
  hub:    array(CITIES) of mpvar
  open:   array(CITIES) of mpvar

  Revenue, Cost, Profit: linctr
end-declarations
```

04 / DATA LOADING

All parameters are loaded from the external f4hubs.dat file. This keeps the model code clean and makes scenario analysis a parameter change, not a code edit.

```
initializations from "f4hubs.dat"
  QUANT DIST FACTOR NAMES OPEN
end-initializations

! Set binary domains
forall(i in CITIES) do
  open(i) is_binary
  hub(i)  is_binary
end-do
forall(i,j in CITIES | i<j) direct(i,j) is_binary
```

05 / STRUCTURAL CONSTRAINTS

Six core constraints enforce the business rules: base airports stay open, hubs must be at open airports, exactly 3 hubs, at least 2 new airports open, and total opening spend within budget.

```
! Base airports always open
forall(i in BASE) open(i) = 1

! Hubs only at open cities
forall(i in CITIES) hub(i) <= open(i)

! At least 2 new airports opened
sum(i in NEW) open(i) >= MIN_NEW_OPEN

! Exactly 3 hubs across the network
sum(i in CITIES) hub(i) = NHUBS

! Budget ceiling for new city openings
sum(i in NEW) OPEN(i) * open(i) <= BUDGET_OPEN
```

06 / DYNAMIC FLOW VARIABLE CREATION

Flow variables are only created for valid routing patterns: intra-US via a single US hub, intra-EU via a single EU hub, and intercontinental US–EU pairs via one hub on each side.

```
! Intra-US: one US hub
forall(i,j,k in US | i<j) create(flow(i,j,k,k))

! Intra-EU: one EU hub
forall(i,j,k in EU | i<j) create(flow(i,j,k,k))

! Intercontinental: US hub k + EU hub l
forall(i,k in US, j,l in EU) create(flow(i,j,k,l))
```

07 / ROUTING CONSTRAINTS

For each open city pair (i,j), exactly one routing must be chosen — either direct or via hubs. If either city is closed, no route is used. Flows are only permitted through cities designated as hubs.

```
forall(i,j in CITIES | i<j) do
  sum(k,l in CITIES | exists(flow(i,j,k,l)))
    flow(i,j,k,l) + direct(i,j) <= open(i)
  sum(k,l in CITIES | exists(flow(i,j,k,l)))
    flow(i,j,k,l) + direct(i,j) <= open(j)
  sum(k,l in CITIES | exists(flow(i,j,k,l)))
    flow(i,j,k,l) + direct(i,j) >= open(i) + open(j) - 1
end-do

! Flows only through chosen hubs
forall(i,j,k,l in CITIES | exists(flow(i,j,k,l))) do
  flow(i,j,k,l) <= hub(k)
  flow(i,j,k,l) <= hub(l)
end-do
forall(i,j,k,l in CITIES | exists(flow(i,j,k,l)))
  flow(i,j,k,l) is_binary
```

08 / OBJECTIVE FUNCTION

Profit = Revenue – TransportCost – EnvironmentalCost – OpeningCost. Revenue is earned for every served OD pair at domestic or international rate per mile. Transport cost includes distance plus the emissions penalty per flight leg. Opening cost is subtracted only for NEW airports.

```
Revenue :=
  sum(i,j in CITIES | i<j)
    QUANT(i,j) * calc_revenue(i,j) *
    ( direct(i,j) + sum(k,l in CITIES |
      exists(flow(i,j,k,l))) flow(i,j,k,l) )

Cost :=
  sum(i,j,k,l in CITIES | exists(flow(i,j,k,l)))
    QUANT(i,j) * calc_cost(i,j,k,l) * flow(i,j,k,l)
  + sum(i,j in CITIES | i<j)
    QUANT(i,j) * calc_cost_direct(i,j) * direct(i,j)

Profit := Revenue - Cost - sum(i in NEW) OPEN(i) * open(i)

maximize(Profit)
```

09 / COST & REVENUE FUNCTIONS

calc_cost handles the two routing cases: one hub (10% distance discount, 2 flights) and two hubs (20% discount on hub-to-hub leg only, 3 flights). An emissions penalty is added per flight. calc_revenue differentiates domestic vs international revenue per mile.

```
  function calc_cost(i,j,k,l: integer): real
    declarations
      base: real; flights: real
    end-declarations
    if k = l then
      ! One hub: i -> k -> j  (10% discount, 2 flights)
      base    := ONE_HUB_FACTOR * (DIST(i,k) + DIST(k,j))
      flights := 2.0
    else
      ! Two hubs: i -> k -> l -> j  (20% hub-hub, 3 flights)
      base    := DIST(i,k) + FACTOR*DIST(k,l) + DIST(l,j)
      flights := 3.0
    end-if
    returned := base + EMISS * flights
  end-function

  function calc_cost_direct(i,j: integer): real
    returned := DIST(i,j) + EMISS * 1.0
  end-function

  function calc_revenue(i,j: integer): real
    if ((i in US) and (j in US)) or ((i in EU) and (j in EU)) then
      returned := REV_DOM * DIST(i,j)   ! Domestic rate
    else
      returned := REV_INT * DIST(i,j)   ! International rate
    end-if
  end-function

end-model
```

References

- Guéret, C., Prins, C. and Sevaux, M. (2002). *Applications of Optimization with Xpress-MP*. Dash Optimization.
- Alumur, S. and Kara, B.Y. (2008). Network Hub Location Problems: The State of the Art. *European Journal of Operational Research*, 190(1), pp. 1–21.