Methodology, Modelling, and Consulting Skills
School of Mathematics
The University of Edinburgh
Year: 2022/2023

THE UNIVERSITY *of* EDINBURGH

Computer Session 3: More Xpress techniques

This handout gives you more Xpress techniques to work with.

1 Integer variables

To mark a variable as integer or binary, you introduce a new constraint. The two constraints are `is_integer` or `is_binary`.

For example, you can write for an array of variables `is_producing` indexed by a set `periods`:

```
forall(t in periods)
  is_producing(t) is_binary
```

Lets solve the knapsack problem that we discussed in class,

$$\begin{array}{ll} \max & \sum_{i=1}^{4} b_i x_i \\ \text{s.t.} & \sum_{i=1}^{4} x_i w_i \leq W \end{array}$$

with the following data

Item	1	2	3	4
b_i	16	22	12	8
w_i	5	7	4	3

and the total available space in my knapsack is $W = 14$. Our Morsel model will look as follows.

```
model Knapsack

uses "mmxprs"

declarations

number_of_items = 4
Items = 1..4
benefit, weight: array(Items) of integer
Item_names: array(Items) of string
```

```
x: array(Items) of mpvar

end-declarations
! popolate data
initialisations from "knapsack.dat"
benefit weight Item_names
end-initialisations

! Objective function
Total_benefit := sum(i in Items)benefit(i)*x(i)

! Capacity constraint
sum(i in Items) weight(i)*x(i) <= 14

! Binary requirements
forall(i in Items)
x(i) is_binary

maximize(Total_benefit)

forall(i in Items) if(getsol(x(i))=1) then writeln("* Pack your ",Item_names(i),"
if
writeln
forall(i in Items) writeln("Your ", Item_names(i), " utilises ",
strfmt(100*getsol(x(i))*weight(i)/14,0,2),"% of your knapsack.")
```

2 Problem status

When Xpress solves a problem, the information on the status (optimal solution, unbounded problem, infeasible problem) is stored in the internal parameter `getprobstat`. Some of the values are: `XPRS_OPT` if the solver has finished with an optimal solution and `XPRS_INF` if the problem is infeasible. For example, we could write the following:

```
if getprobstat = XPRS_OPT then
    writeln("Problem solved to optimality.")
elif getprobstat = XPRS_INF then
    writeln("Problem is infeasible.")
end-if
```

If the model is solved to optimality, the first condition of the `if` will be true. If the model is infeasible, the second condition will be true.

3 Procedures

A procedure is a block of code that is written independently and then referred to by a given name. It is useful if we want to structure our code clearly and/or if we are using this block several times. For example, we can improve the output when looking at the status by creating a procedure with the name `DisplaySolution`. First, we write this line after the initialisations block:

```
forward procedure DisplaySolution
```

This line says that, later in the code, we will define a procedure named `DisplaySolution`. However, since it has already been declared, we can call it before having defined it explicitly. This line is not necessary if we define the procedure before it is used for the first time, but it is considered good practice to define procedures at the end of the code.

Now, in order to display the information, we write after the `maximize` line the following:

```
if getprobstat = XPRS_OPT then
    DisplaySolution
elif getprobstat = XPRS_INF then
    writeln("Problem is infeasible.")
end-if
```

The advantage of writing the code like this is that the structure is nicer. We can understand more clearly this block where possible solution statuses are checked and leave for later the details on how to display the solution in case the problem is solved to optimality.

Finally, we write the full procedure:

```
procedure DisplaySolution
writeln("Problem solved to optimality.")
writeln
writeln("Optimal knapsack benefit: ",getobjval,".")
writeln
forall(i in Items)
if(getsol(x(i))=1) then
writeln("* Pack your ",Item_names(i),".");
end-if
writeln
forall(i in Items)
writeln("Your ", Item_names(i), " utilises ",
strfmt(100*getsol(x(i))*weight(i)/14,0,2),"% of your knapsack.")
end-procedure
```