Linear Programming Problems with Discrete Variables

Strictly Speaking

- If a formal model has variables that may only take specific (discrete) values, then it is ***not a linear programming problem***
- It will be an "integer programming problem," or a "mixed integer-linear programming problem," or sometimes a "binary integer programming problem"

When Do You Need Discrete Variables?

- To model some aspects of continuous problems
- When there are "yes/no" aspects to the decision (or other logical relationships)
- When the decision is "how many" rather than "how much"

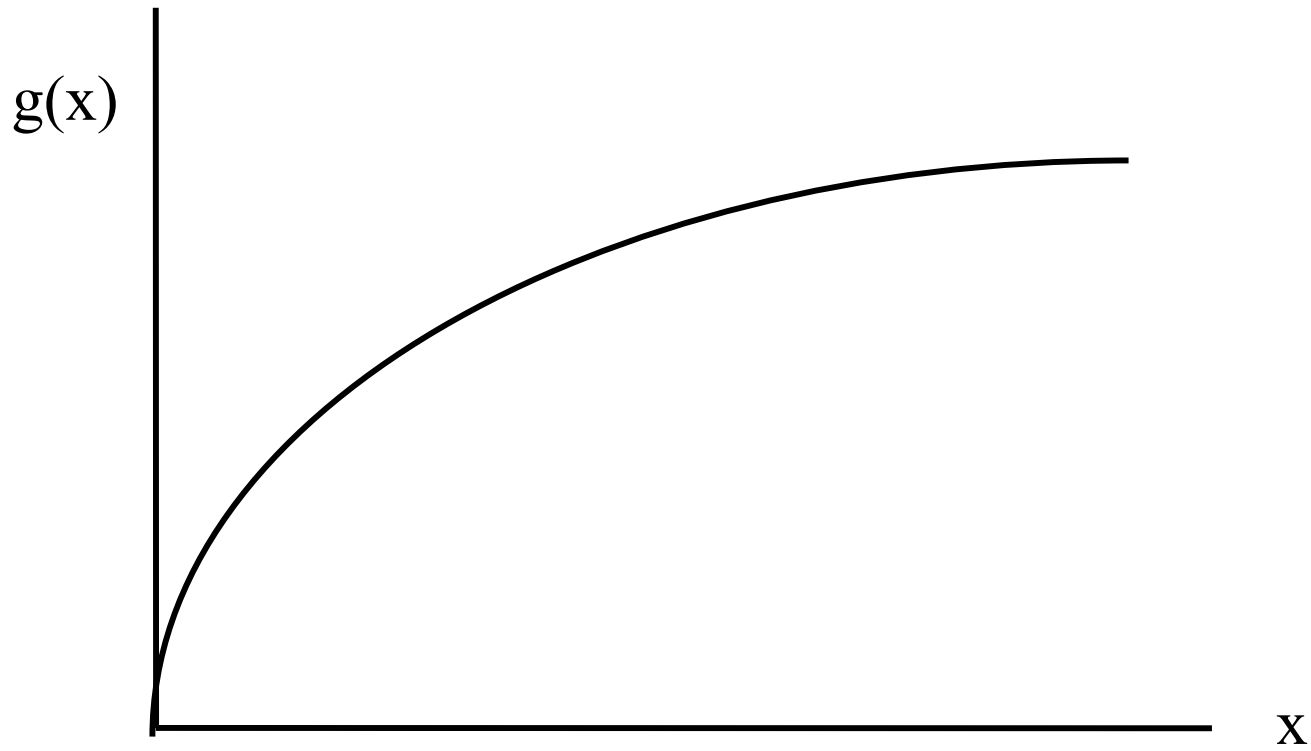
Mosel Allows Discrete Variables

Specify the type of an “mpvar” in a “constraint”, e.g.,

forall (j in 1..n) x(j) is_integer OR forall (j in 1..n) x(j) is_binary

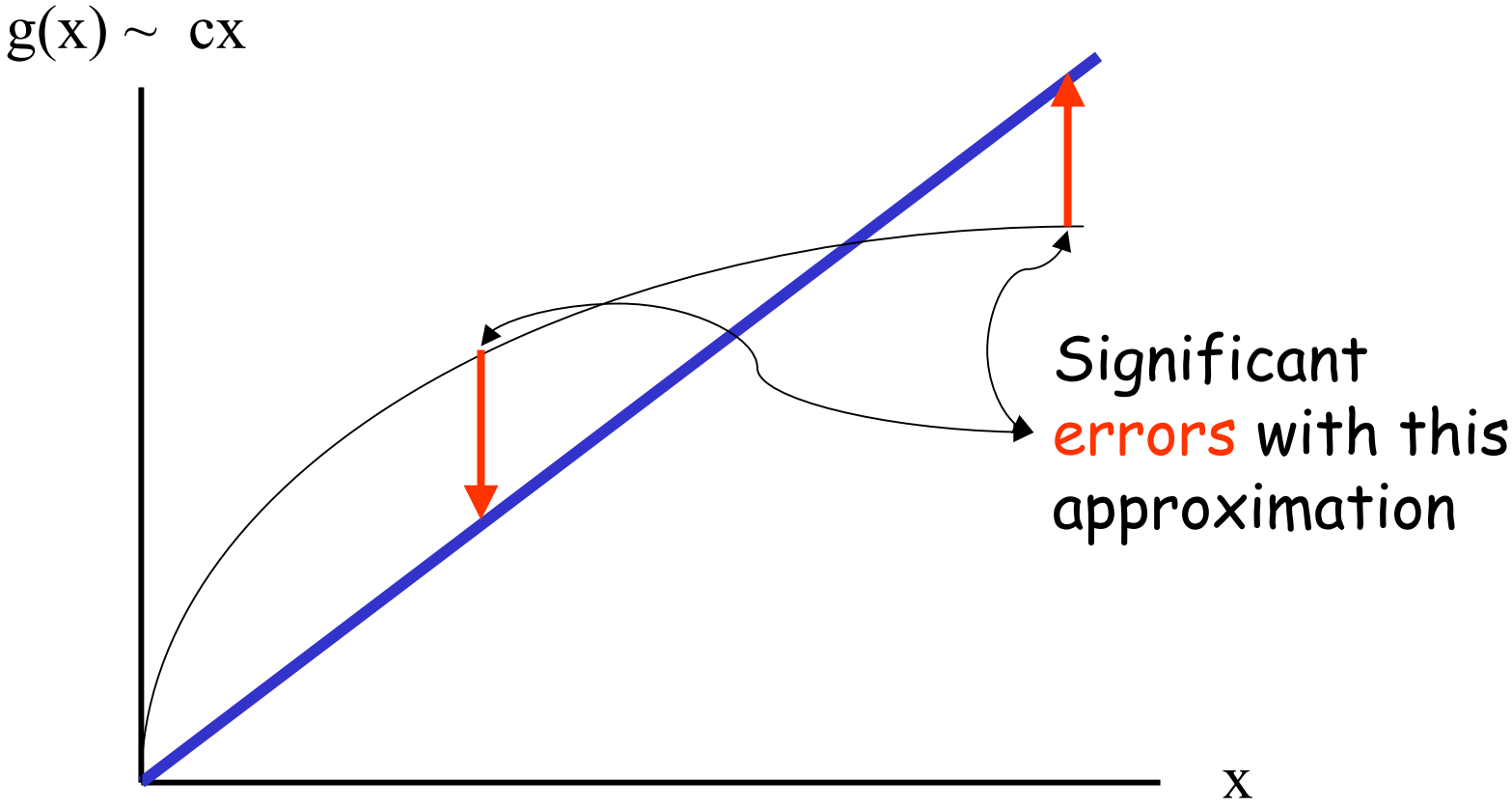
Integer variables may only take integer values between 0 and the upper bound. The largest upper bound for any integer variable is the optimiser parameter INTMAX. By default, this is set to the largest permissible 32 bit integer, 2147483647. Integer variables with an upper bound of unity are treated as binary variables.
Binary variables may only take the values 0 or 1. Sometimes called 0/1 variables.

Example: Nonlinear Costs



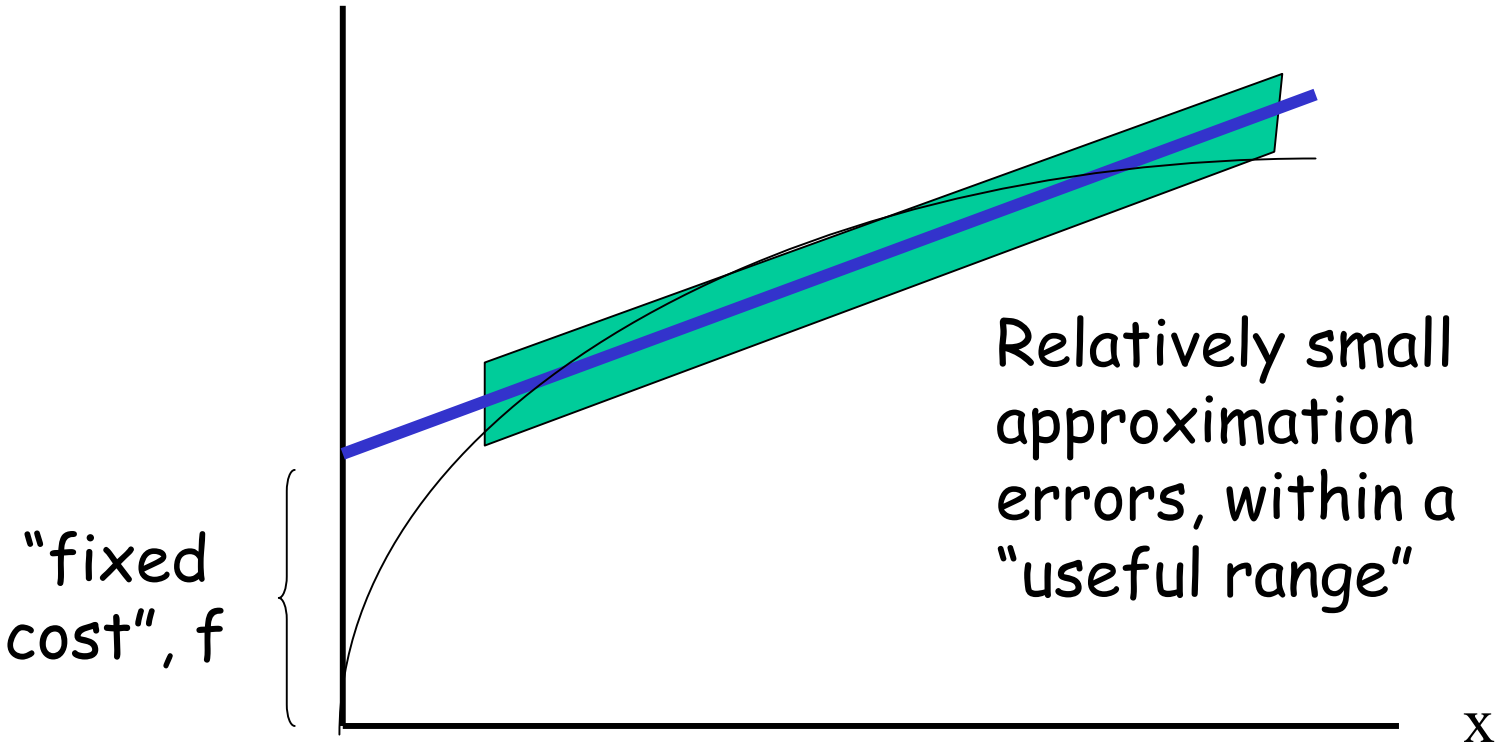
"economies of scale"

Modeling Cost Functions in LP

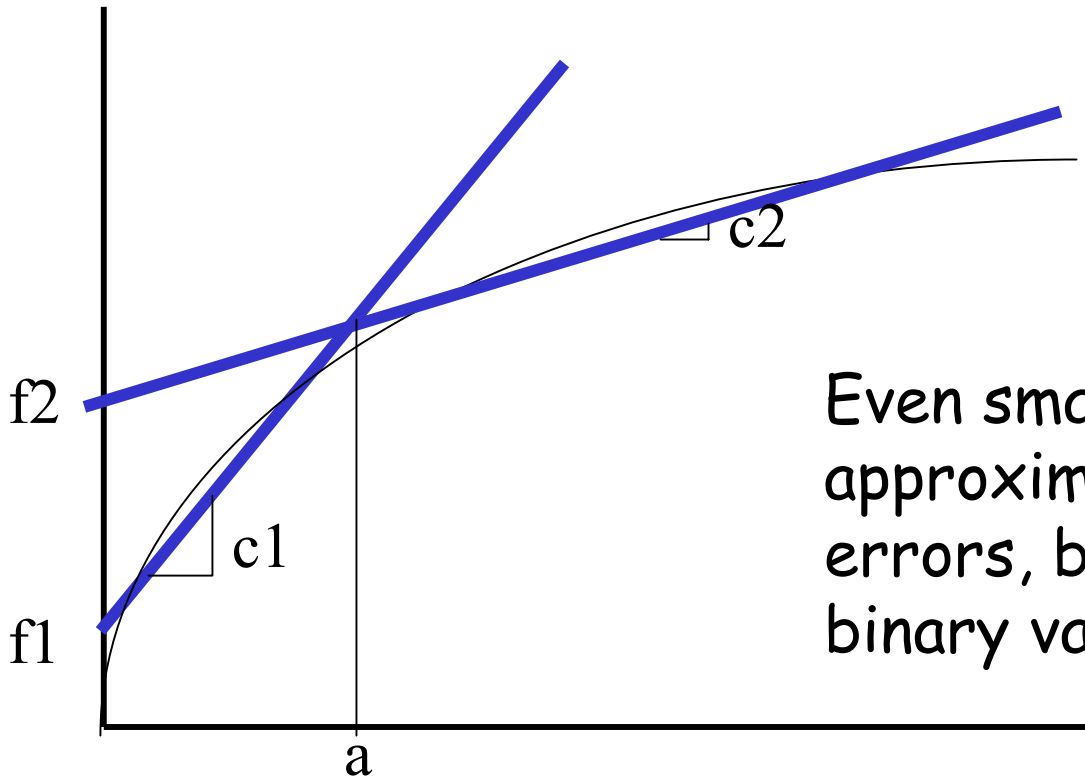


Different Approximation

$$g(x) \sim f\,y + c\,x, \text{ where } y = \begin{cases} 0, \text{ if } x = 0 \\ 1, \text{ if } x > 0 \end{cases}$$



Different Approximation



Even smaller approximation errors, but more binary variables

How to Model?

There is a non-obvious "trick" to doing this!
Treat each linear function as a separate cost function, with its own "variable"

$$x <= x1 + x2$$

Associate a binary variable with each of the linear functions:

$$y1 = \begin{cases} 0, \text{ if } x1 = 0 \\ 1, \text{ if } x1 > 0 \end{cases}$$

$$y2 = \begin{cases} 0, \text{ if } x2 = 0 \\ 1, \text{ if } x2 > 0 \end{cases}$$

Enforce the requirement that you can't use x2 unless y1=1 (TRICK!)

$$y2 \leq y1$$

Then

$$g(x) = f1y1 + (f2-f1)y2 + c1x1 + c2x2$$

Why does this work?

- It costs something to use x2
- Since you are minimizing, you will never use x2 unless doing so costs less
- Since you must allow the use of x1 (y1=1) before you can use x2, if all you need of x is less than a, it's cheaper to use x1

Why show this to you?

- It's a clever modeling trick, and a good example of what you can do if you know the tricks
- It's not obvious, and most of us (myself included) would have a hard time "inventing" this ourselves

Including Fixed Costs in LP Models

- Define the 0-1 variable
- Use the 0-1 variable to force the fixed cost to be included in the objective function
- Add a constraint to force the 0-1 variable to be 1 if x is non-zero

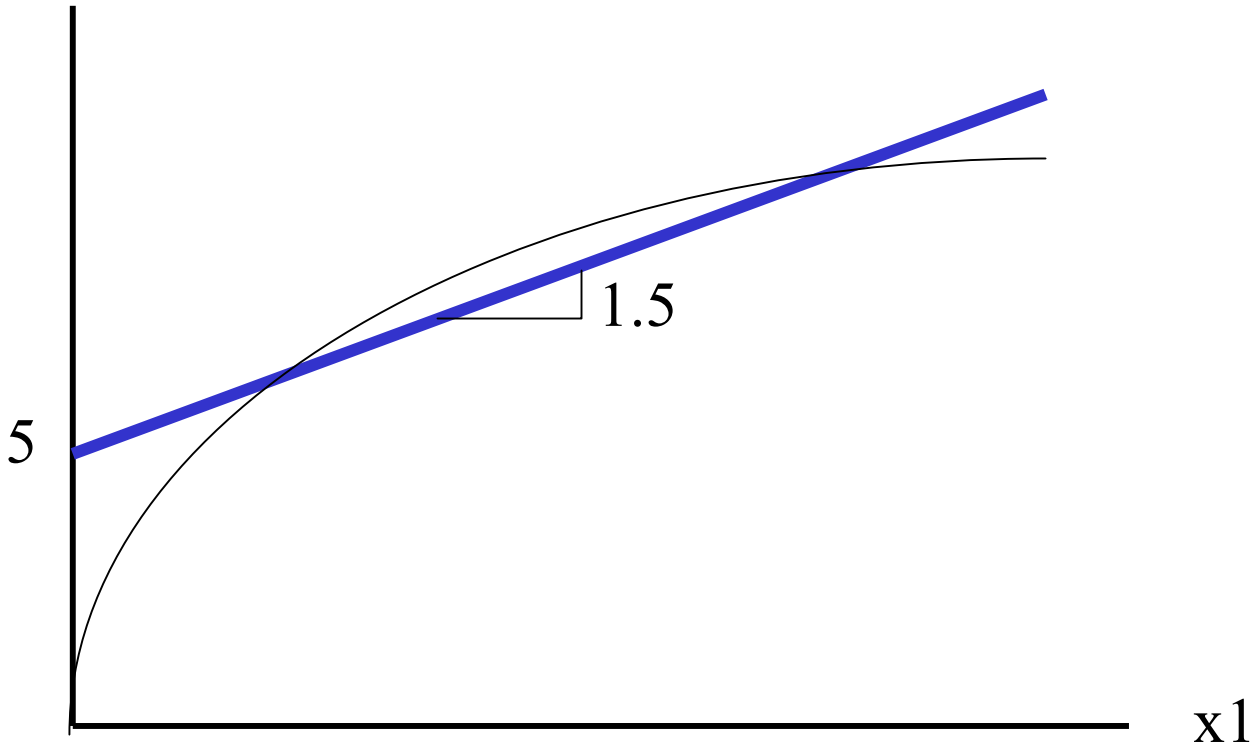
$$x - M y <= 0$$

where M is larger than x can possibly be

Example

- Suppose in the Leary Chemical problem, the cost to run process number 1 is really non-linear

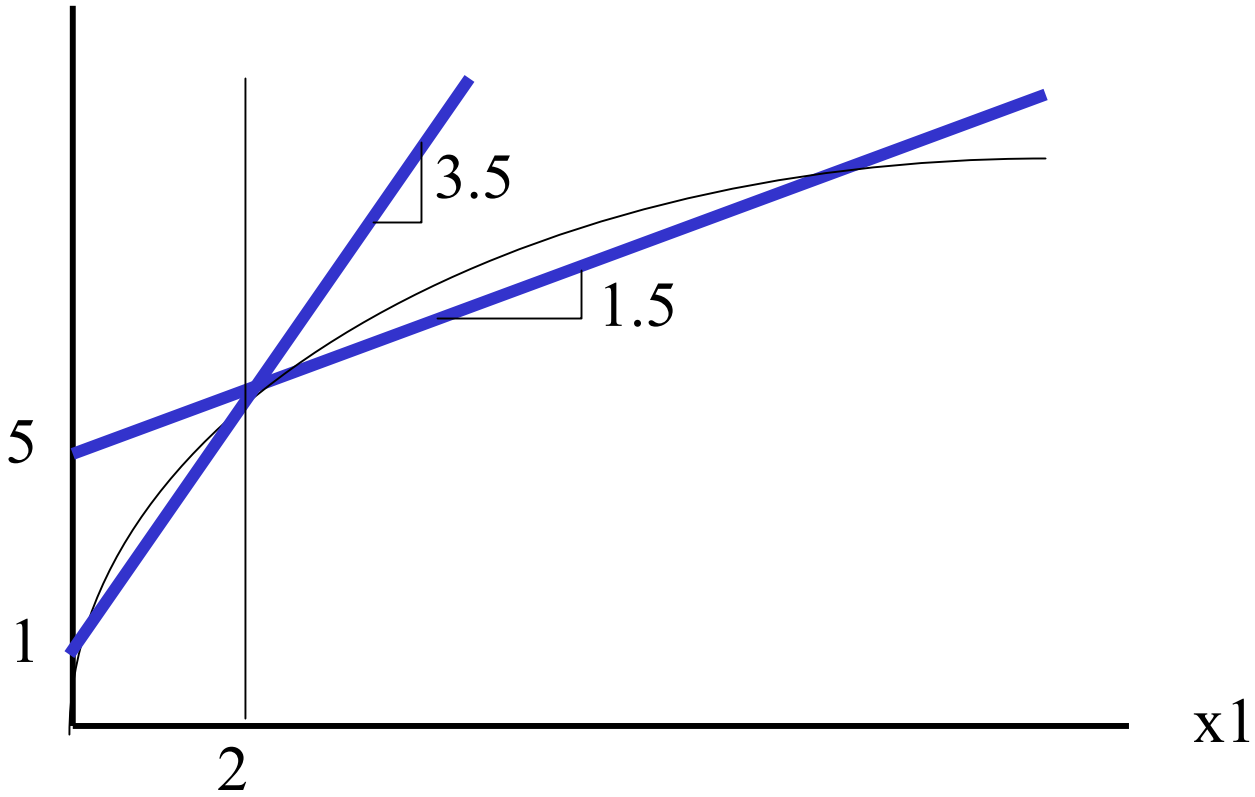
Leary, Process 1 Cost



Revised Model (learyFC.mos)

```
model Leary                                  ! Start a new model
uses "mmxprs"                                ! Load the optimizer library
declarations
x1: mpvar                                    ! number of trucks
x2: mpvar
y: mpvar                                     ! number of cars
end-declarations
! Objective: maximize total profit
cost:= 5*y +1.5*x1 + x2
! constraints
prodA:= 3*x1 + x2 >= 10
prodB:= x1 + x2 >= 5
prodC:= x1      >= 3
fix:=  x1 -10*y <= 0
y is_binary
minimize(cost)                               ! Solve the LP-problem
```

Leary, Process 1 Cost



Revised Model (exercise!)

Assignment

- Suppose Farmer Jones has a "setup cost" for both corn and wheat
- Modify the LP model to incorporate the setup costs
- Solve the revised model using Xpress (try different values for the setup costs to see if you can force different solutions to be optimum)

Archetypes, yet again

- Associating a binary variable with a continuous variable in order to enforce some logical requirement
- In this example, the logical requirement is value relative to a parameter
- Could be value relative to another continuous variable

Examples

- Can't use any of material ***b*** until you have used all of material ***a***
- Can't produce both product ***1*** and product ***2*** in the same period
- The minimum hours per week is 20

Examples

*Can't use any of material **b** until you have used all of material **a***

- $x1 \leq a1$ and $x2 \leq a2$
- let y2 be binary
- $a1-x1 \leq M*(1-y2)$, and $x2 \leq M*y2$
- if $a1-x1 > 0$, then $y2=0$, o/w $y2>0$ is ok

Examples

*Can't produce both product **1** and product **2** in the same period*

- Let yj = 1 if you produce product j (e.g., $x_j \leq M^* y_j$)
- Then $y1 + y2 \leq 1$ insures that you produce at most one product type

Examples

The minimum hours per week is 20

- Let x be the hours per week
- Let y1=1 if x=0 and y2=1 if x $\geq$ 20
- y1 + y2 = 1
  x $\leq$ M*(1-y1)
  x $\geq$ 20*y2

There are problems in which the decisions are naturally discrete, or integer valued

Making Change Problem

- You make a purchase, and your cost is $11.06. You give the clerk a $100 bill.
- What is the minimum number of pieces of money that the clerk can give you to make your change (=$88.94)?
- In $US, this problem is trivial and can be solved in your head! (what is the algorithm?)
- Can also formulate it as an optimization problem

Formulation

- "Coins" have values: $0.01, $0.05, …, $5, $10, $20, and $50
- Let x(i) be the number of coins of type i used to make change

Formal Model

$$\min \quad \sum_{i=1}^{n} x_i$$

$$s.t. \quad \sum_{i=1}^{n} v_i x_i = change$$

$$x_i \in I, \; i = 1,\ldots,n$$

This model has only one constraint! It is "small".

Mosel Model

```
MODEL coins
uses "mmxprs"                ! Load the optimizer library
declarations
   NumCoins = 11
   Amount = 88.94
   Value: array(1..NumCoins) of real
   x: array(1..NumCoins) of mpvar
end-declarations
   Value:= [50.0, 20.0, 10.0, 5.0, 2.0, 1.0, 0.50, 0.25, 0.10, 0.05, 0.01]
MinCoins:=     sum(i in 1..NumCoins) x(i)
MakingChange:= sum(i in 1..NumCoins) Value(i) * x(i) = Amount
forall (i in 1..NumCoins) x(i) is_integer
minimize (MinCoins)

writeln("Solution status is:  ", getprobstat, "\n Amount of change is:  ", Amount)
forall (i in 1..NumCoins) writeln(Value(i), " number: ", getsol(x(i)))

end-model
```

Archetypes, again

- The making change problem is a variation of the "knapsack" archetype:

$$optimize \quad \sum_{i=1}^{n} c_i x_i$$

$$s.t. \quad \sum_{i=1}^{n} v_i x_i <=> cons$$

$$x_i \in I, \ i = 1,...,n$$

$$or$$

$$x_i \in \{0,1\}, \ i = 1,...,n$$

Knapsack Problems ...

- are very ***easy*** as discrete optimization problems go
- Even so, they are much harder to solve (in terms of computational effort) than linear programming problems of similar dimensions (number of variables)

Assignment

- Download the coins.mos file and run it for different values of "amount"
- Observe how long it takes to solve the problem
- Modify coins.mod to add $100 bill, and run it for amount=988.94

Generalization

- Discrete optimization problems can be very difficult to solve, in fact, you may not be able to compute an optimum solution for a specific problem, if it is "too large"
- Use caution when considering formulation that involve discrete variables

Where Do You Need Them?

- Selecting among investment alternatives
- Designing supply chains
- Sourcing products
- Production/inventory planning
- Crew scheduling
- Vehicle routing

Expectations

- You can formulate simple LP and mixed integer-LP problems
- You can solve 2-variable problems graphically
- You can use Visual Xpress to solve simple problems
- You can use indexed variables and data files in Visual Xpress