CS 524: Introduction to Optimization
Lecture 6 : Multi-Period Models

Michael Ferris

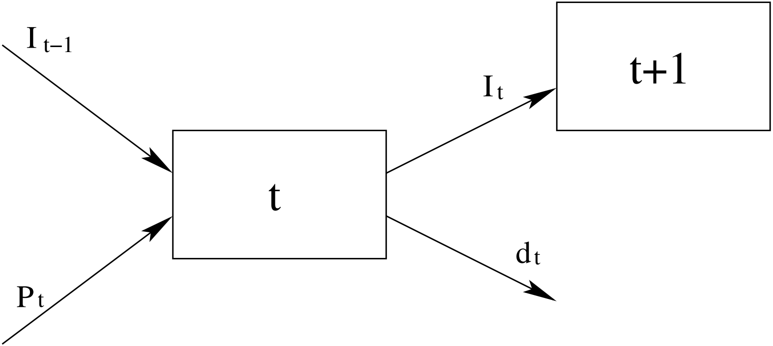
Computer Sciences Department
University of Wisconsin-Madison

September 18, 2023

Modeling Multi-Period Problems

- One of the most important uses of optimization is in multi-period planning.
- Partition time into a number of periods.
- Usually distinguished by Inventory or Carry-Over variables.
- Suppose there is a "planning horizon" $T = \{1, 2, \ldots, |T|\}$.
- Also suppose there is a known demand d_t for each $t \in T$
- Define...
  - P_t : Production level in period $t, \forall t \in T$
  - I_t : Inventory level in period $t, \forall t \in T$

Modeling Multi-Period Problems



$$I_1 = I_0 + P_1 - d_1$$
$$I_2 = I_1 + P_2 - d_2$$
$$I_t = I_{t-1} + P_t - d_t$$

- To model "losses or gains", just put appropriate multipliers (not 1) on the arcs

Lags and Leads in GAMS

- GAMS has lag and lead operators: `+,-`
- These are distinguished/different from arithmetic operators $(+, -)$ by the context of the code
- You can use these to get the next or previous element in the set
- Note: A reference to a non-existent element causes the default value (zero) to be used,
- An attempt to assign to a non-existent element results in no assignment being made.
- Warning: You won't get an error either!
- This is an important "feature"[1] – we'll use it in our models

[1] not a bug!

GAMS Interlude: Ordered Sets

- One-dimensional sets have an ordering associated with their elements: the order in which they are specified. (See "Ordered Sets" section of the GAMS User Guide)
- The compile time directive `$onuellist` produces listing of all unique set elements, in the order specified and in some alphanumerically sorted order
- The function `card(I)` gives the number of elements in `I`
- For a one-dimensional set `I`, `ord(I)` or `I.ord` returns the order of the element in set `I`.
- Example: `ordered_sets.gms`
- Note that if you change the elements of `I`, then ordering goes away (as it does if elements are not ordered in global ordering above)

Another Use of ord() - see ordered_sets.gms

- Population in year 2000 is 56,000,000,
- Growth rate 1.5% per annum. Calculate population in years 2001-2020.
- We use "**" operator for exponentiation

```
set years /2000*2020/;
parameter population(years);
population(years) = 56*(1.015**(ord(years)-1));
display population;
```

Alternative (setname.ord):

```
set years /2000*2020/;
parameter population(years);
population(years) = 56*(1.015**(years.ord-1));
display population;
```

Another Story: Aggregate Planning

- Complex production process involving many pieces
  - Demands
  - Variable workforce size
  - Overtime possibilities
  - Inventory requirements

We're Making Shoes: ShoeCo

- Plan production of shoes for next several months
- Meet forecast demands on time
- Hire and/or lay off workers
- Make overtime decisions
- Objective: minimize total cost

ShoeCo: It's All Greek To Me

- Planning horizon $T = \{1, 2, \ldots |T|\}$. ($|T| = 4$).
- Meet demand d_t for shoes in period $t \in T$.
  $d = (3000, 5000, 2000, 1000)$
- Initial Shoe Inventory: $\mathcal{I}_0 = 500$
- Have $\mathcal{W}_0 = 100$ workers currently employed
- Workers paid $\$\alpha = 1500$/month for working $H = 160$ hours
- They can work overtime (max of $O = 20$ hours/worker) and get paid $\$\beta = 13$/hour.

ShoeCo: Greek Letter Zoo

- It take $a = 4$ hours of labor and $\delta = \$15$ in raw material costs to produce a shoe
- Hire-Fire costs: $\eta = 1600$ to hire a worker, $\zeta = \$2000$ to fire a worker.
- Running out of greek letters, $\iota = \$3$ holding cost incurred for each pair of shoes held at the end of the month.

Your Mission

- Minimize all costs: labor (regular + overtime), production, inventory, hiring and firing
- What decision variables do we need?
  - HINT: If you're having trouble getting the decision variables, try and write the objective

Decision Variables

- x_t: # of shoes to produce during month t
- I_t: Ending inventory in month t, $t \in T \cup \{0\}$
- w_t: # of workers available in month t, $t \in T \cup \{0\}$.
- o_t: # of overtime hours used in month t
- h_t: # workers hired at the beginning of month t
- f_t: # workers fired at the beginning of month t

Objective, Minimize Total Costs

1. Raw Material Costs: $\sum_{t \in T} \delta x_t$
2. Regular Labor Costs: $\sum_{t \in T} \alpha w_t$
3. Overtime Labor Costs: $\sum_{t \in T} \beta o_t$
4. Hiring Costs: $\sum_{t \in T} \eta h_t$
5. Firing Costs: $\sum_{t \in T} \zeta f_t$
6. Inventory Costs: $\sum_{t \in T} \iota I_t$

$$\min \sum_{t \in T} (\delta x_t + \alpha w_t + \beta o_t + \eta h_t + \zeta f_t + \iota I_t)$$

Constraints

Limit on Monthly Production

- Not given explicitly
- Determined by number of workers available and overtime decisions
- Math-speak: $ax_t \leq Hw_t + o_t \quad \forall t \in T$

Upper limit on overtime hours/month

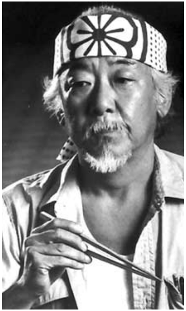
- Depends on how many workers you have
- Aggregate planning: Don't worry about individual workers
- Math-speak: $o_t \leq Ow_t \quad \forall t \in T$

Constraints

Demand must be met on time

- Equivalent to having nonnegative ending inventory each month (no backlogging)
- Math-speak: $I_t \geq 0 \quad \forall t \in T$
- This assumes we have balance between production, demand, and inventory
- We'll see backlogging later

Balance, Daniel-Son



Shoes

- Draw Picture, Math Speak:

$$I_t = I_{t-1} + x_t - d_t \quad \forall t \in T$$

- Boundary: $I_0 = \mathcal{I}_0$ (Maybe $I_{|T|} \geq \mathcal{I}_0$).

People

- Hiring/Firing Affects worker levels. Math speak:

$$w_t = w_{t-1} + h_t - f_t \quad \forall t \in T$$

- Boundary: $w_0 = \mathcal{W}_0$

Full Model

$$\min \sum_{t \in T} (\delta x_t + \alpha w_t + \beta o_t + \eta h_t + \zeta f_t + \iota I_t)$$

s.t.

$$\begin{aligned}
a x_t &\leq H w_t + o_t \quad \forall t \in T \\
o_t &\leq O w_t \quad \forall t \in T \\
I_t &= I_{t-1} + x_t - d_t \quad \forall t \in T \\
I_0 &= \mathcal{I}_0 \\
w_t &= w_{t-1} + h_t - f_t \quad \forall t \in T \\
w_0 &= \mathcal{W}_0 \\
x_t, I_t, w_t, h_t, f_t, o_t &\geq 0 \quad \forall t \in T
\end{aligned}$$

GAMS and Logical Conditions (See Gams User Guide)

- A numerical expression is "false" if 0, "true" otherwise.
  - `2a-4` is false if a=2, true otherwise.
- Set membership. If we have sets `days` and subset `workdays(days)`
  - `workdays(days)` evaluates to true if that element of `days` belongs to `workdays`, false otherwise.
- Numerical relations. e.g. `(A < B)` where `A` and `B` are numbers.
  - Operators on numbers are intuitive: `<, <=, =, <>, >=, >`.
  - Note: Different than constraint operators `=L=, =E=, =G=`.
- Logical operators: `not, and, or, xor`.
- Logical conditions have a numerical value: 1 for true and 0 for false.

`sameas()`: see `sameas.gms`

- If `I` is a set and `elt` may or may not be an element of `I`, then `sameas(‘‘elt’’,I)` evaluates to true if `elt` belongs to `I`, false otherwise.
- You may wish to do conditional processing depending on text defining the name of a set element
- `sameas(setelement,othersetelement)` returns true if the text giving the name of `setelement` is the same as the text for `othersetelement` and a false otherwise.
- `sameas(asetelement,"texttotest")` returns true if the text giving the name of `asetelement` is the same as the `‘‘texttotest’’` and false otherwise.

```
Set i / Beijing, Calcutta, Mumbai, Sydney, Joburg, Cairo /;
Set j / Rome, Paris, Boston, Cairo, Munich, Calcutta, Barca /;
```

GAMS and the $

- **Dollar Condition.** Use conditional expressions in combination with "$" to define conditional statements.
- See "Dollar Control Options" of the User's Guide.

```
Scalar  cntSameC;
cntSameC = sum((i,j)$sameas(i,j),1);
```

Dollar on the right

- Syntax: `(parameter or variable attribute) = (numerical expression)$(conditional expression);` always makes an assignment.
- If `conditional expression` evaluates to true, then `(numerical expression)` is assigned to `(parameter or variable attribute)`. Otherwise zero is assigned to `(parameter or variable attribute)`.

Holla' for the Dolla'

- Can combine terms in "dollar on the right" e.g.

```
A(I) = 1$(ord(I) <= 2) + X(I-2)$(ord(I) > 2)
```

sets `A(I)` to 1 for the first two elements of `I` and to `X(I-2)` for the other elements of *I*.

Using It In ShoeCo

- `set T /Jan, Feb, Mar, Apr/ ;`
- `I(T) =E= I0$(ord(T) eq 1) + I(T-1) + x(T) - d(T) ;`
  - Note: Use of `$`.
  - Use of "out of range" index

Exercise

- Download `shoeco.gms`
- Review model slides from lecture to populate data of this model (into file `shoeco.inc`)
- Run and ensure the optimal cost is $ 692,500
- Review the compile time listing to review whether the equations are generated correctly
- Pay particular attention to the `cost_eq`, `BalShoe_eq`, and `BalPeople_eq`
- Note whether the initial time periods and final time periods have the correct entries
- Explain why you don't fire 25 people in April
- Save the file for later use with a backlogging extension

Dolla' on the Left

- Syntax: `(parameter or variable attribute)$(conditional expression) = (numerical expression);` means that if `conditional expression` evaluates to true, then `(numerical expression)` is assigned to `(parameter or variable attribute)`.
- Otherwise no assignment is attempted (that is, the value of `(parameter or variable attribute)` is not altered).
- Ex: `A(I)$(ord(I) <2) = B`, `A(days)$weekday(days) = B`.

More GAMS Logical Stuff

- The functions ord and card are frequently used to single out the first or last element of an ordered set.
- For example, we may want to fix a variable for the first and last elements of a set:
  x.fx(i) $ (ord(i) = 1) = 3;
  x.fx(i) $ (ord(i) = card(i)) = 7;
- Alternative:
  x.fx(i) $ (i.first) = 3;
  x.fx(i) $ (i.last) = 7;
- Also have the notion of a *singleton* set (a set with at most one element)
  singleton set j(i);
  j(i) = yes$i.first; x.fx(j) = 3;
  j(i) = yes$i.last; x.fx(j) = 7;