CS 524: Introduction to Optimization
Lecture 11 : Sailco and profiling

Michael Ferris

Computer Sciences Department
University of Wisconsin-Madison

September 29, 2023

Review

- Using multi-dimensional sets - `payoff.gms` (model $(j, l) \in M(t)$)
- Minimum cost example - `mincost2.gms` (loop and display)
- Shortest path example - `short2.gms` (seed, loop and alias)

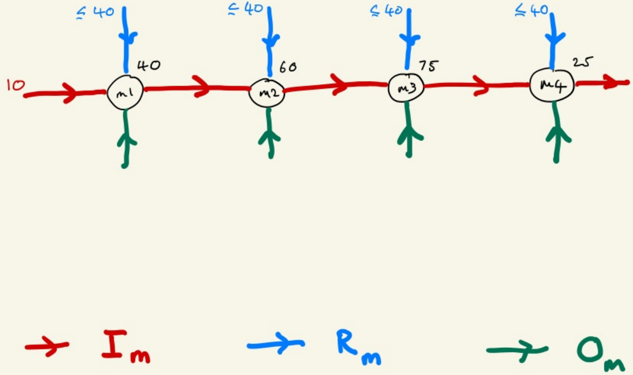
Come Sail Away, Come Sail Away

- Sailco manufactures sailboats. During the next 4 months the company must meet the following demands for their sailboats:
  - 40, 60, 75, 25
- At the beginning of Month 1, Sailco has 10 boats in inventory.
- Each month it must determine how many boats to produce.

Sailco, Cont.

- During any month, Sailco can produce up to 40 boats with regular labor and an unlimited number of boats with overtime labor.
  - Boats produced with regular labor cost $400 each to produce,
  - while boats produced with overtime labor cost $450 each.
- It costs $20 to hold a boat in inventory from one month to the next. Find the production and inventory schedule that minimizes the cost of meeting the next 4 months' demands on time.

Sailco as inventory flow (`sailco.gms`)



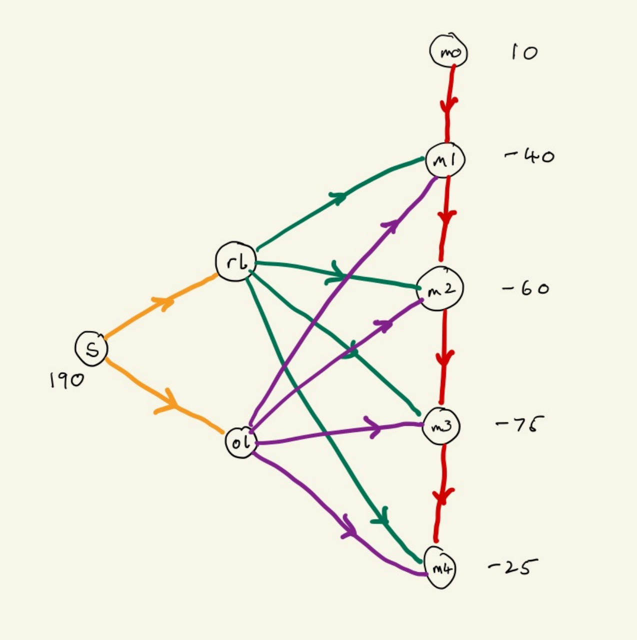
Sailco

- Applications that are not obviously network problems can be modeled as such.

Model as a network (MCNF)

- One node per month – given demand at each month.
- Inflow to each month node from a "regular labor" node, with capacity 40 and cost 400
- Inflow to each month node from a "overtime labor" node, with unlimited capacity and cost 450
- Additional node for sailboats needed (to split into ones produced regular and overtime)
- Month-to-month arcs representing inventory, (cost 20)
- Balance constraints applied at all nodes.

Sailco as MCNF (`sailco2.gms`)



Exercise (sailco3.gms)

- Note in sailco2.gms that we convert data given as “T” to our set “Months” by hand: often necessary to do such data transformations
- Replace all `Months.first` with a singleton set `fMonth`
- Change initial supply to 30
- Change initial supply to N(10,4)

BIG Problems: bigassign.gms

- For large problems, we need to exploit the sparsity. Only define variables and equations that you need!

But Wait, there's more!

- Setting random seeds
- Creating options files within GAMS
- Setting options to solvers
- Different types of LP solvers
- Getting model attributes
- Practice with conditional processing

More GAMS Stuff – Bigassign.gms

1. `$onecho` is done before code run, so if you want multiple option files, you need to use `model.optfile=2`, and name option file `solver.op2`
2. Setting seed to random number generator
   - `option seed=666`
   - `execseed = 1 + gmillisec(jnow);` – Takes from clock
   - `option seed = %gams.user1%;` – Takes from command line `gams model.gms user1=666`
3. See different ways to use `ARCS` set
4. `option bratio=1.0`
   - Doesn't use "advance start" information from previous solves

Full list in "Solver Related Options" of GUG

Model Attributes Controlled by the User

- `iterlim` iteration limit
- `limcol` number of columns displayed for each block of variables
- `limrow` number of rows displayed for each block of equations
- `optfile` option file usage
- `reslim` time limit for solver. Usually in CPU seconds
- `solprint` solution print option

What's taking so long? (GAMS Command Line Parameters)

- Why are things taking so long? What's the bottleneck?
- Can use GAMS/Studio "Parameter Editor" (Icon: $\substack{\vee \\ \vee}$)
- gams filename profile=1 stepsum=1

Timing Attributes Controlled by the User

- `profile` timing of individual statements
- `stepsum` summary of times in each block (first column is time in each block since last summary, second column is cumulative)

"Solver Related Options" in GUG

Some Model Attributes Controlled by the Solver

- `iterusd`: number of iterations used
- `modelstat`: model status (see below)
- `numequ`: number of single equations generated
- `numvar`: number of single variables generated
- `resusd`: resource units (in CPU seconds) used to solve model
- `solvestat`: solver status (see below)

- Actual values for `modelstat` and `solvestat` given in previous lecture (and also in GUG)

Smax

- We often would like to know the largest element in a container that is indexed over a set
- `smax(I,a(I))` returns the largest element of *a*
- There is also `smin`