CS 524: Introduction to Optimization
Lecture 10 : Min cost flows / dynamic sets

Michael Ferris

Computer Sciences Department
University of Wisconsin-Madison

September 27, 2023

GAMS is a two pass system

- GAMS is a two pass system: (I) Compilation and (II) Execution
- Compilation looks over the whole file and determines whether the syntax is correct
- Statements such as assignments (or "solve") are checked for syntactical correctness but not actually executed
- There are special commands that are delineated by the line starts with a "$" - these are compile time directives and are processed at compile time
- Examples include `$ontext`, `$offtext`, `$onecho`, `$offecho`, `$title`, `$call`, `$load`

Static and Dynamic Sets: (section in GUG)

- Static sets (and subsets): membership does not change. (Usually declared at top of file, defined at compile time)
- Dynamic sets are declared in the same way—but should always be declared as a subset of static set, to allow domain checking.
- Initial contents of the dynamic set can be defined (statically) in the same way as for a static subset.
  - Membership can subsequently be altered by direct assignment.
- Some use and misuse examples (or gotcha's) are provided in the file `dynamic.gms`

Examples: see file `dynamic-simple.gms`

- Use of assignment statements such as

```
subitem1('ink') = yes; subitem2('perfume') = no;
```

  turn membership of individual elements on/off.
- Turn on all the elements in the set `firm` for the second index at once:

```
supply('pen',firm) = yes;
```

  This is also possible using a static declaration (set.firm or #firm)

```
set supply(stuff,firm) / pencil.bic, pen.set.firm /;
```

  or

```
set supply(stuff,firm) / pencil.bic, pen.#firm /;
```

- Using the $ for conditional assignment to dynamic sets

Why we May Care—Sparsity of Models

Defining equations over the domain of dynamic sets

- Can't *declare* equation over a dynamic set
- Trick is to use the static superset of the dynamic set to declare the equation, then define it with the dynamic set.
- Can also use dynamic set to change the items you are putting into a sum from one model run to another

- Could have been useful for "coffee tables" in Homework 2, Q2.

Declaring and Defining Items: see dynamic-simple.gms

```
set allr /n,s,w,e,n-e,s-w/
    r(allr);
scalar price /10/;
equations prodbal(allr);
variables activity(allr)
  revenue(allr);

* equation prodbal declared over allr, defined over r
prodbal(r).. activity(r)*price =e= revenue(r);

* sum over the dynamic set only
parameter totinv, inventory(item);
inventory(item) = ord(item);
totinv = sum(item$subitem1(item), inventory(item));

* alternative for the same thing
totinv = sum(subitem1, inventory(subitem1));
```

Multi-dimensional Sets: see `multi_sets.gms`

- In the models we build, we often will need a mapping between elements of different sets
- Maps are built using (multiply-indexed) sets

```
* two basic sets
set months /Jan, Feb, Mar, Apr, May, Jun, Jul, Aug,
            Sep, Oct, Nov, Dec/;
set weather /wintry, spring-like, summery, fall-like/;

* subset with domain checking
set evenMonths(months) / Feb, Apr, Jun, Aug, Oct, Dec/;

* month-weather associations with domain checking
set
        likely_weather (months, weather) /
                      (Nov, Dec, Jan, Feb, Mar, Apr).wintry,
                      (Mar, Apr, May, Jun).spring-like,
                      (May, Jun, Jul, Aug, Sep, Oct).summery,
                      Sep.fall-like, Oct.fall-like, Nov.fall-like/;
```

New Problem!

- Given (directed) network $G = (N, A)$
- Each node $i \in N$ has a "supply" b_i
- $b_i < 0 \Rightarrow$ node i "demands" an amount b_i.
- Arcs $a \in A$ may have costs c_a or capacities u_a
- All demand must be met exactly $\Rightarrow$ for feasibility, we must have that $\sum_{i \in N} b_i = 0$. (Supply = Demand)
  - We can often add a "dummy" node to account for this

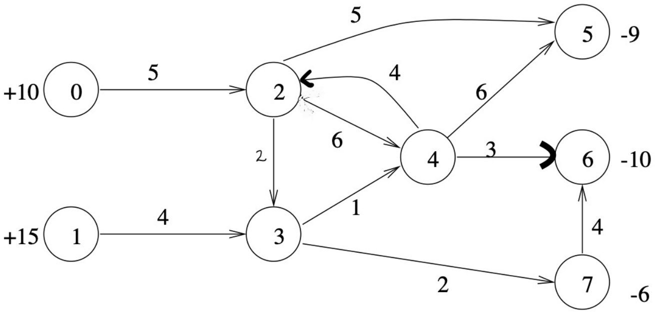
Min-Cost Network Flow

Find a minimum cost flow of the commodity from supply nodes to demand nodes without exceeded the arc capacity

MCNF: Mathematical Formulation

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\begin{aligned} \sum_{k:(i,k)\in A} x_{ik} - \sum_{j:(j,i)\in A} x_{ji} &= b_i \quad \forall i \in N \\ 0 \le x_{ij} &\le u_{ij} \quad \forall (i,j) \in A \end{aligned}$$



MCNF: Mathematical Formulation

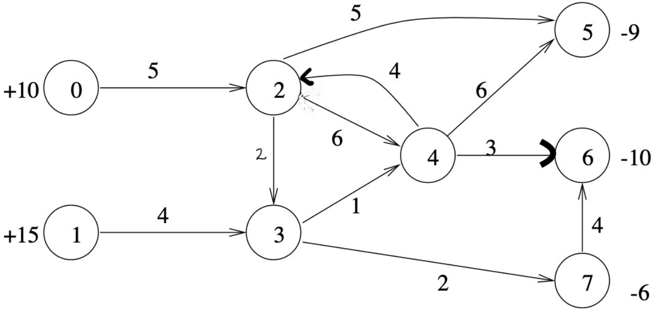
$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\begin{aligned} \sum_{k:(i,k)\in A} x_{ik} - \sum_{j:(j,i)\in A} x_{ji} &= b_i \quad \forall i \in N \\ 0 \le x_{ij} &\le u_{ij} \quad \forall (i,j) \in A \end{aligned}$$

$$\min c^T x$$

$$\begin{aligned} \mathcal{A}x &= b \\ 0 &\le x \le u \end{aligned}$$

$\mathcal{A}$ is the node-arc incidence matrix we have seen before



mincost2.gms

GAMS time: `mincost.gms`, `mincost2.gms`

- `abort` : this allows you to check inputs and stop if they are not good!
- Important: Note use of dynamic set `arc`

- You can loop over sets in GAMS. `loop`
- Important GAMS attributes: `modelstat, solvestat`
- Note use of (two column) list option in `display x.l` statement

Assignment = MCNF

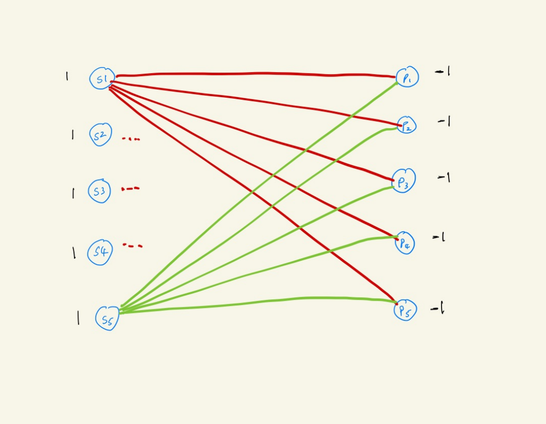
Another MCNF Sample

You can also model the assignment problem as Min-Cost Network Flow

- n nodes: students
- n nodes: projects
- n^2 arcs: linking student node to project node
- Students: supply 1
- Projects: demand 1

- There must be an integer solution, so we need not restrict x_{ij} to be binary

Assignment as MCNF (see `assignprefs1.gms`)



$$\max \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\begin{aligned} \sum_{j \in J} x_{ij} &= 1 \quad \forall i \in I \\ -\sum_{i \in I} x_{ij} &= -1 \quad \forall j \in J \\ 0 \le x_{ij} &\le 1 \quad \forall i \in I, j \in J \end{aligned}$$

Shortest Path Problem

Shortest Path Problem

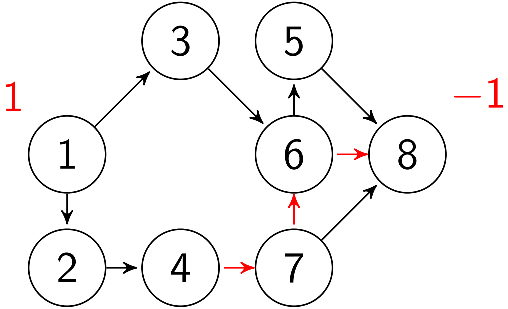
Find the shortest path through a network from a specified origin to a specified destination.

- This is a special case of min cost flow, where costs are distances, there is a single supply source with supply $+1$ and a single demand (sink)

- There are specialized algorithms for shortest path that do not exploit the relationship with min-cost flow and in fact are more efficient from the complexity viewpoint: $O(|A| + |N| \log |N|)$

Example: SPP as MCNF (see `short1.gms`)

- Data for MCNF: nodes, arcs (obvious), c, b, u
- b has a single $+1$ (origin) and a single -1 (destination)
- u_{ij} can either be 1 or ∞ (doesn't matter)
- c_{ij} are typically the distances (note `sqr` and `sqrt`)



Red arcs are lake crossings and have "double" the cost

Extension: Average length random shortest path

- `short2.gms`
  - Note: `option seed=`
  - Note: `loop` (This is cool)
  - Note: difference between assignments to arcs(i,j) and arcs(i,i). Use of the `alias` is essential!
  - Note: `modelstat`

Count the CPU Time:

- `totalSolveTime = totalSolveTime + short.resusd;`