Lecture Outline

• Network flow problems
• Problem 1: Maximum flow problem
• Ford Fulkerson algorithm
• Problem 2: Shortest path routing
• Bellman Ford algorithm
• Simple IP routing: RIP
• Dynamic Programming
Graph Theory Notation

$G = (V, E)$: directed graph with vertex set $V$ and edge set $E$

$b_i$: external supply to each node $i \in V$

$u_{ij}$: capacity of each edge $(i, j) \in E$

$c_{ij}$: cost per unit flow on edge $(i, j) \in E$

$I(i) = \{ j \in V | (j, i) \in E \}$: set of start nodes of incoming edges to $i$

$O(i) = \{ j \in V | (i, j) \in E \}$: set of end nodes of outgoing edges from $i$

Sources: $\{ i | b_i > 0 \}$. Sinks: $\{ i | b_i < 0 \}$

Feasible flow $f$:

- Flow conservation: $b_i + \sum_{j \in I(i)} f_{ji} = \sum_{j \in O(i)} f_{ij}$, $\forall i \in V$
- Capacity constraint: $0 \leq f_{ij} \leq u_{ij}$
### Basic Formulation

Network flow problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{(i,j) \in E} c_{ij} f_{ij} \\
\text{subject to} & \quad b_i + \sum_{j \in I(i)} f_{ji} = \sum_{j \in O(i)} f_{ij}, \quad \forall i \in V \\
& \quad 0 \leq f_{ij} \leq u_{ij}
\end{align*}
\]

In matrix notation as a LP:

\[
\begin{align*}
\text{minimize} & \quad c^T f \\
\text{subject to} & \quad Af = b \\
& \quad 0 \leq f \leq u
\end{align*}
\]

where \( A \in \mathbb{R}^{\mid V \mid \times \mid E \mid} \) is defined as

\[
A_{ik} = \begin{cases} 
1, & \text{i is the start node of edge k} \\
-1, & \text{i is the end node of edge k} \\
0, & \text{otherwise}
\end{cases}
\]
Special Cases

- Maximum flow problem (this lecture)
- Shortest path problem (this lecture)
- Transportation problem (uncapacitated bipartite graph)

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j} c_{ij} f_{ij} \\
\text{subject to} & \quad \sum_{i=1}^{m} f_{ij} = d_j, \quad j = 1, \ldots, n \\
& \quad \sum_{j=1}^{n} f_{ij} = s_i, \quad i = 1, \ldots, m \\
& \quad f_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n
\end{align*}
\]

Variables $f_{ij}$. Constants $d_j, s_i, c_{ij}$

- Assignment problem (homework):

$m = n, d_j = s_i = 1$ in transportation problem
Maximum Flow Problem

maximize \( b_s \)
subject to \( A f = b \)
\( b_t = -b_s \)
\( b_i = 0, \ \forall i \neq s, t \)
\( 0 \leq f_{ij} \leq u_{ij} \)

Reformulated as network flow problem:

- Costs for all edges are zero
- Introduce a new edge \((t, s)\) with infinite capacity and cost \(-1\)
- Minimize total cost is equivalent to maximize \( f_{ts} \)
**Ford Fulkerson Algorithm**

1. Start with feasible flow $f$
2. Search for an augmenting path $P$
3. Terminate if no augmenting path
4. Otherwise, if flow can be pushed, push $\delta(P)$ units of flow along $P$ and repeat Step 2
5. Otherwise, terminate

Q: How to find augmenting path?
Q: How much flow can be pushed?
Augmenting Path

Idea: find a path where we can increase flow along every forward edge and decrease flow along backward edge by the same amount. Still satisfy constraints. Increase objective function

Augmenting path: a path from $s$ to $t$ such that $f_{ij} < u_{ij}$ on forward edges and $f_{ij} > 0$ on backward edges.

Augmenting flow amount along augmenting path $P$:

$$\delta(P) = \min \left\{ \min_{(i,j) \in F} (u_{ij} - f_{ij}), \min_{(i,j) \in B} f_{ij} \right\}$$

Can search for augmenting path by following possible paths leading from $s$ and checking conditions above.
Example
Example
Max Flow Min Cut Theorem

Theorem: If optimal value is finite, Ford Fulkerson algorithm terminates with an optimal flow

Theorem: If edge capacities $u_{ij}$ are integers, edge flow variables remain integer

Definition: cut $S$ is a subset of $V$ such that $s \in S$ and $t \notin S$

Definition: capacity of cut $C(S)$ is sum of edge capacities on edges that cross from $S$ to its complement:

$$C(S) = \sum_{(i,j) \in E \mid i \in S, j \notin S} u_{ij}$$

Theorem: Value of maximum flow $\max b_s$ equals minimum cut capacity $\min_S C(S)$
Shortest Path Routing

Given a directed graph with vertex set $V$ and edge set $E$
Each edge $(i, j)$ has cost or length $c_{ij}$
Allow negative length edges, but no negative length cycles

Our development follows DP algorithm
Other approaches (e.g., duality) and algorithms (e.g., Dijkstra) possible

Consider all-to-one shortest path routing with destination vertex $n$
**Bellman Ford Algorithm**

Let $p_i(t)$ be length of shortest path from $i$ to $n$ using at most $t$ edges, with $p_i(t) = \infty$ if no such path exists.

Let $p_n(t) = 0$, $\forall t$ and $p_i(0) = \infty$, $\forall i \neq n$.

$p_i(t + 1)$ consists of two parts:

- cost of getting from $i$ to a neighboring $k$.
- cost of getting from $k$ to destination $n$.

Pick the minimum total cost:

$$p_i(t + 1) = \min_{k \in \mathcal{O}(i)} \{c_{ik} + p_k(t)\}$$
Example
Example
Example

```
0 -> 2
|   6 |
|     |
0 -> 7
|   7 |
0 -> 8
|   2 |
7 -> 2
|   9 |
2 -> 4
|  -3 |
2 -> 5
|  -2 |
4 -> 7
|   7 |
4 -> 2
|  -4 |
```
Example
IP Routing

Basic versions:

- **IGP (e.g., RIP)**: distance-vector based
- **IGP (e.g., OSPF, IS-IS)**: link-state based
- **EGP (e.g., BGP4)**: across Autonomous Systems

Extensions:

- Multicast routing
- Mobile IP
- Mobile wireless ad hoc routing
- QoS routing
RIP Routing

Simple example (homework):

Practical concerns:
- Loop avoidance
- Stability
- Speed of convergence
- Scalability
Sequential Optimization

Additive cost in discrete time dynamic system:

\[ x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, \ldots, N - 1 \]

State: \( x_k \in S_k \)

Control: \( u_k \in U_k(x_k) \)

Random disturbance: \( w_k \in D_k \) with distribution conditional on \( x_k, u_k \)

Admissible policies:

\[ \pi = \{\mu_0, \ldots, \mu_{N-1}\} \]

where \( \mu_k(x_k) = u_k \) such that \( \mu_k(x_k) \in U_k(x_k) \) for all \( x_k \in S_k \)

Given cost functions \( g_k, k = 0, \ldots, N \), expected cost of \( \pi \) starting at \( x_0 \):

\[
J_\pi(x_0) = \mathbb{E} \left( g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right)
\]

Optimal policy \( \pi^* \) minimizes \( J \) over all admissible \( \pi \), with optimal cost:

\[
J^*(x_0) = J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)
\]
**Principle of Optimality**

Given optimal policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$. Consider subproblem where at time $i$ and state $x_i$, minimize cost-to-go function from time $i$ to $N$:

$$E \left( g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right)$$

Then truncated optimal policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$ is optimal for subproblem.

Tail of an optimal policy is also optimal for tail of the problem.
**DP Algorithm**

For every initial state $x_0$, $J^*(x_0)$ equals $J_0(x_0)$, the last step of the following backward iteration:

\[
J_N(x_N) = g_N(x_N)
\]

\[
J_k(x_k) = \min_{u_k \in U_k(x_k)} E\left( g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right), \quad k = 0, \ldots, N-1
\]

If $\mu_k^*(x_k) = u_k^*$ are the minimizers of $J_k(x_k)$ for each $x_k$ and $k$, then policy

\[
\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}
\]

is optimal

Proof: induction and Principle of Optimality
Deterministic Finite-State DP

- No stochastic perturbation:
  \[ x_{k+1} = f_k(x_k, \mu_k(x_k)) \]

- Finite state space: \( S_k \) are finite for all \( k \)

Deterministic finite-state DP is equivalent to shortest path problem in trellis diagram
Lecture Summary

- Network flow problems are special cases of LP that model a wide range of problems in networking and problems modelled by graphs.

- Maximum flow problems and shortest path problems are two important special cases of network flow problems that can be efficiently solved by special purpose distributed algorithms.

- DP principle is extremely powerful for sequential optimization.

- We will later study powerful generalizations of Network, Flow Problems to Network Utility Maximization.

- Practical issues in IP routing (IGP and BGP) to be taught in Rexford guest lecture.

Reading: Section 7.1, 7.2, 7.5, and 7.9 in Bertsimas and Tsitsiklis