# Advanced Algorithms

Floriano Zini

Free University of Bozen-Bolzano
Faculty of Computer Science

Academic Year 2013-2014

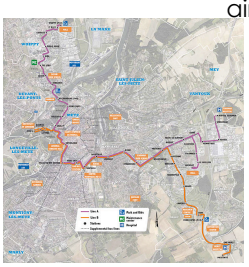
# Lecture 8 – Network Optimization Algorithms

# Introduction

- **Network models** have numerous **practical applications**. E.g.:

electrical networks

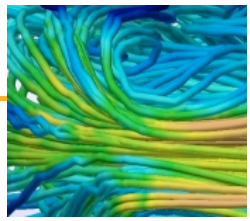

airline crew scheduling



transportation

water distribution

# Introduction (cont.)



- In all these cases we have a **Flow Capacity Problem**
  - One or more **commodities** are **flowing** from one point to another through a **network** whose **branches** have various **constraints** and **flow capacities**
  - The **direction of flow** in each branch and the **capacity** of each branch are **known**

The problem is to **determine the maximum flow,** of the network

# Basic definitions

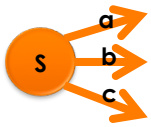
- A **network** is a (directed) **graph**
- The **edges** have an associated **flow**
- Some **examples**:

| Nodes | Edges | Flow |
|---|---|---|
| Intersections | Roads | Vehicles |
| Airports | Air lanes | Aircraft |
| Switching points | Wires, channels | Messages |
| Pumping stations | Pipes | Fluids |

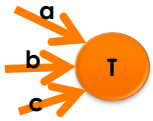
- Each **edge** may have a **capacity** that is an **upper** (and sometimes **lower**) **bound on the amount of flow that can be carried along that edge**
  - E.g., the maximum water flow in a pipe or the minimum water flow to prevent condensation

# Basic definitions (cont.)

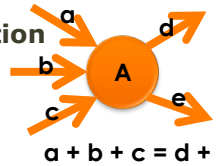
- A **source node** (or supply node) is a node which **introduces flow into a network**




- A **sink node** (or demand node) is a node which **removes flow from a network**




- A **transshipment node** satisfies the **conservation** of the flow, i.e., **flow in equals flow out**

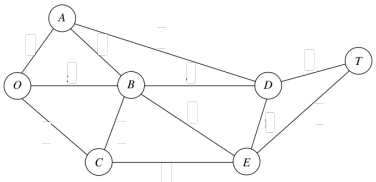



a + b + c = d + e

- **Sinks** and **sources** model **entities** located **at the boundary** between the **network** under study and the **external world**

## An example – National Park

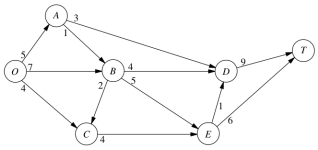
- A **limited amount** of sightseeing is requested for **nature preservation**
  - Cars are not allowed
  - Tourists are transported by trams driven by the park rangers
- The **road system** in the park is represented by the following **graph**




  - ***O*** is the **entrance** of the park
  - ***T*** is a **scenic wonder** to be visited by tourists
  - **Other letters** indicate locations of **ranger stations**
- A **small number of trams** are used to **transport tourists from** the entrance ***O*** to the scenic wonder ***T*** and back

## An example – National Park (cont.)

- Trams usually go from *O* to *T* along the shortest route
- But, during the **peak season, more people** want to take the tram to *T*
  - To increase the number of daily tram trips **various routes might be followed**
  - To avoid disturbing the ecology and wildlife, a strict **bound** has been placed on the **number of tram trips that can be made on each of the roads per day**



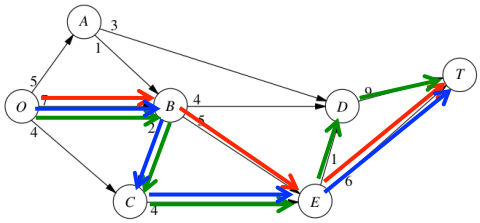

Numbers on edges are the capacity of the road, i.e. the maximum number of trams per day

- The problem is **how to route the various trips to maximize the number of trips that can be made per day without violating the limits on individual roads**


**How to route the various trips to maximize the number of trips that can be made per day without violating the limits on individual roads?**

**Can you spot a (not necessarily optimal) feasible solution?**

- One **feasible solution** is to send **7 trams per day**
  - 5 along the route $O \rightarrow B \rightarrow E \rightarrow T$
  - 1 using $O \rightarrow B \rightarrow C \rightarrow E \rightarrow T$
  - 1 along the route $O \rightarrow B \rightarrow C \rightarrow E \rightarrow D \rightarrow T$
- There exists a better solution?
- This kind of problem is called **maximum flow problem**




## Maximum flow problem

- **Definition**
  - Given a network (a directed graph) $G = (V , E)$ **the maximum flow problem is to find a feasible flow from the single source of *G* to the single sink of *G* having the maximum value**
- The **network** is subjected to some **restrictions**
  - It must contain **one source node** and **one sink node**
  - All the **remaining nodes** are **transshipment nodes**
  - **Flow** through an **edge** is allowed **only in the direction indicated by the arrow** and the **maximum amount of flow** is given by the **edge capacity**
- The objective is to **maximize the total amount of flow from the source to the sink**
  - **This amount** is measured either as **the amount leaving the source** or as **the amount entering in the sink**

## Maximum flow problem (cont.)

Some **applications** of the maximum flow problem are

- Maximize the **flow of goods** through a company's distribution network from its factories to its customers
- Maximize the **flow of oil** through a system of pipelines
- Maximize the **flow of water** through a system of aqueducts
- Maximize the **flow of vehicles** through a transportation network
  - E.g., traffic engineers may want to know the maximum **flow rate of vehicles from the downtown car park to the freeway on-ramp** because this will influence their decision on whether to widen the roadways
- A telephone company wants to know the maximum **number of simultaneous calls between two cities** via the various land-lines, satellites and microwave towers it can support

## Maximum flow problem (cont.)

- There is a function to maximize subject to constraints…. Looks familiar?
- **Yes! The maximum flow problem can be modeled using linear programming!**
- Take the National Park problem
  - **Variables**
    - tram trips per day on each of the roads in the park
  - **Objective function**
    - maximize the total number of trips per day
  - **Constraints**:
    - **Flow conservation**
      - for each transshipment node, the trams that enter the node also exit the node
    - **Capacity**
      - the number of tram trips cannot exceed the maximum number of trips allows on a road
    - **Non negativity**
      - the number of tram trip per day on road must be positive

## Maximum flow problem in Linear Programming

Formally, a **maximum flow problem**, represented by a graph $G = (V , E)$ is **modeled in LP** as follows:

- **Variables**
  - $x_{ij}$, $(i, j) \in E$
- **Objective function**
  - *maximize* $\sum_{(s, i) \in E} x_{si}$ *(s: source node)* or
  - *maximize* $\sum_{(i, t) \in E} x_{it}$ *(t: sink node)*
- **Constraints**
  - **Flow conservation:**
    $\sum_{(k,i) \in E} x_{ki} = \sum_{(i, j) \in E} x_{ij}$ **for each transshipment node** $i \in V$
  - **Capacity**: $x_{ij} \leq u_{ij} \quad (i, j) \in E$
  - **Non negativity**: $x_{ij} \geq 0 \quad (i, j) \in E$

## Solving the maximum flow problem

- Fortunately ☺, there are also other algorithm for solving the maximum flow problem
- We will see two of them
  - Ford-Fulkerson algorithm
  - Edmonds-Karp algorithm
    - specialization of the Ford-Fulkerson method

## Ford-Fulkerson algorithm - Idea

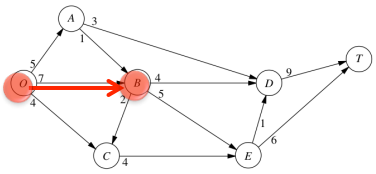
- The Ford-Fulkerson Algorithm was proposed in 1956
- The **key idea** behind the algorithm is
  - **as long as there is a path from the source to the sink, with available capacity on all edges in the path, we send flow along it**
- A path with available capacity is called an **augmenting path**
- In general at each step there are several **alternative augmenting paths**

## Ford-Fulkerson algorithm – Idea (cont.)

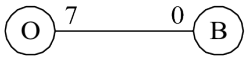
- After a flows has been assigned to the edges in an augmenting path, the **residual network** is computed
- The residual network **brings information about remaining edge capacities** (called residual capacities)
- For building the residual network we consider the undirected graph from the original network and we associate to each edge *(i, j)* two numbers
  - The **first number** represent **the residual capacity from node i to node j**
  - The **second number** represent the **flow allocated from node i to node j** or, equivalently, the **residual capacity from node j to node i**

## Ford-Fulkerson algorithm (cont.)

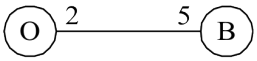
As for the National park problem




- consider the edge *(O,B)* having capacity 7
- The residual network at the first iteration of the algorithm will contain this subnetwork




- If we assign a flow of 5 to *(O,B),* we obtain a residual network containing the subnetwork




This means that up to 2 units of glow can be sent from O to B, but also that the flow from O to be can be decreased up to 5 units

## Ford-Fulkerson algorithm (cont.)

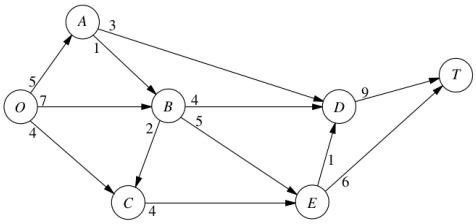
Formally, the Ford-Fulkerson algorithm works in **5 steps**

1. Build the **residual network** starting from the network ***G***
2. Identify a **path from** the **source to** the **sink** such that **each edge of the path has a positive residual capacity**
   - This is an **augmenting path**
   - If **no augmenting path** exists then **STOP**
3. Identify the **residual capacity *c**** of the **augmenting path** and **increase the flow in the path by *c****
   - The **residual capacity** is the **minimum of the residual capacities of the edges in the path**
4. **Update** the **residual capacities** of **each edge *(i,j)*** on the **augmenting path**
   - by ***subtracting c**** to the residual capacity of ***(i,j)***
   - by **adding *c**** to the residual capacity of ***(j,i)***
5. Return to step 2

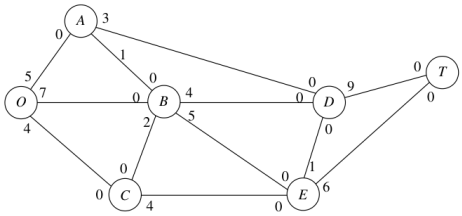
**NB1**: in the selection of the augmenting path (step 2) the graph is considered **undirected**!

## Ford-Fulkerson alg. – National Park problem

Network G



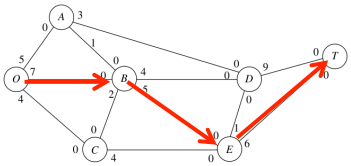

**Step 1**
Initial residual network from G



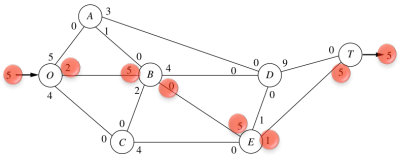

## Ford-Fulkerson alg. – National Park problem (cont.)

**Iteration 1**

- **Step 2**: select an augmenting path
  - $O \rightarrow B \rightarrow E \rightarrow T$
- **Step 3**: identify the residual capacity $c*$ of the augmenting path and increase the flow in this path by $c*$
  - $c* = min\{7,5,6\} = 5$
- **Step 4**: update the residual capacities of each edge on the augmenting path




Residual network

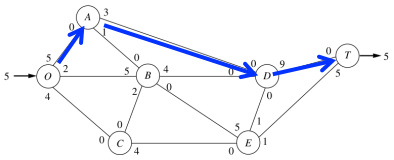



New residual network

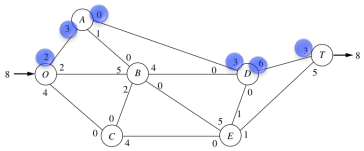
## Ford-Fulkerson alg. – National Park problem (cont.)

**Iteration 2**

- **Step 2**: select an augmenting path
  - $O \rightarrow A \rightarrow D \rightarrow T$
- **Step 3**: identify the residual capacity $c*$ of the augmenting path and increase the flow in this path by $c*$
  - $c* = min\{5,3,9\} = 3$
- **Step 4**: update the residual capacities of each edge on the augmenting path



Residual network

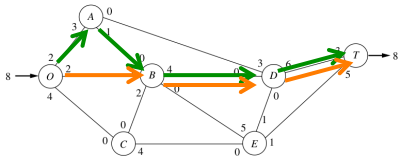


New residual network

## Ford-Fulkerson alg. – National Park problem (cont.)

**Iteration 3**

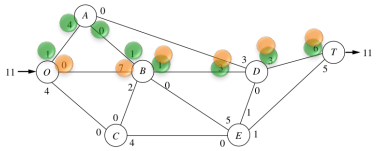
- Assign a flow of 1 to the augmenting path $O \rightarrow A \rightarrow B \rightarrow D \rightarrow T$



Residual network

**Iteration 4**

- Assign a flow of 2 to the augmenting path $O \rightarrow B \rightarrow D \rightarrow T$

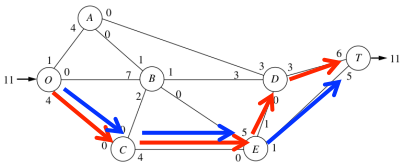


New residual network

## Ford-Fulkerson alg. – National Park problem (cont.)

**Iteration 5**

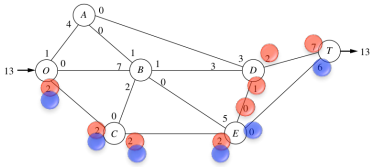
- Assign a flow of 1 to the augmenting path
  - $O \rightarrow C \rightarrow E \rightarrow D \rightarrow T$



Residual network

**Iteration 6**

- Assign a flow of 1 to the augmenting path
  - $O \rightarrow C \rightarrow E \rightarrow T$

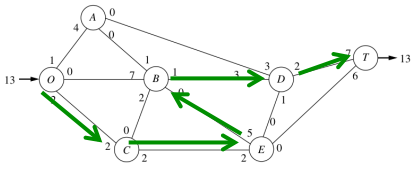


New residual network

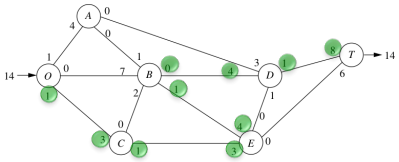
## Ford-Fulkerson alg. – National Park problem (cont.)

**Iteration 7**

- Assign a flow of 1 to the augmenting path
  - $O \rightarrow C \rightarrow E \rightarrow B \rightarrow D \rightarrow T$



Residual network

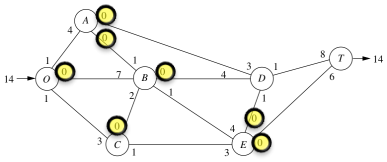


New residual network

**Iteration 8**

- No augmenting paths

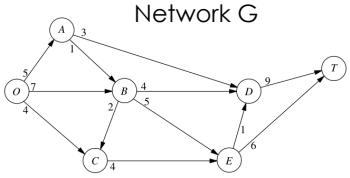
**STOP!!**



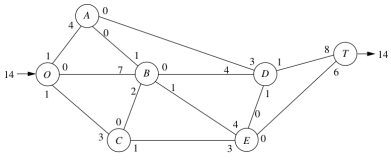
## Ford-Fulkerson alg. – National Park problem (cont.)

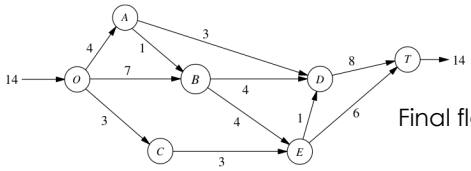
- The **final flow** on each edge is the **flow allocated to the edge** (second number) in the final residual network, or analogously, the difference between the **original edge capacity** and the the **final edge residual capacity** (first number)

Network G




Final residual network




Final flows

## Ford-Fulkerson algorithm – Pseudo code

1. create the residual network *Gr*
2. **while** there is some directed path from *s* to *t* in *Gr* **do**
   1. let *P* be a path from *s* to *t* in *Gr*
   2. $c^* = min_{(ij) \in P} r_{ij}$
   3. send $c^*$ unit of flow along *P*
   4. update $r_{ij}$ for each $(i,j) \in P$

NB: $r_{ij}$ is a pair, including the residual capacities in both directions

## Ford-Fulkerson algorithm – Integral Theorem

- **If each edge in a flow network has integral capacity, then there exists an integral maximal flow**
- Lemma:
  - **At each iteration all residual capacities are integral**
- Proof (by induction on the number *k* of augmentations):
  - k=0: the initial capacities are integral
  - Assume it is true after *k−1* augmentations
  - Consider augmentation *k* along path *P*
    - The residual capacity $c^*$ of *P* is the smallest residual capacity on *P*, which is integral
    - After updating, we modify residual capacities (that are integral) by 0, or $c^*$
    - Thus residual capacities stay integral

## Ford-Fulkerson algorithm – Complexity

- Let all edge capacities be integers
- The **runtime** of Ford-Fulkerson is bounded by ***O(|E|*f)***, where ***|E|*** is the **number of edges** in the graph and ***f*** is the **maximum flow**
- This because **each augmenting path can be found in *O(|E|)*** time and **increases the flow by an integer amount which is at least 1**

# Next lecture

- **The Ford-Fulkerson algorithm does not specify which alternating augmenting path to use if there is more than one**
- In 1972, Edmonds and Karp proposed a **heuristic** for **choosing** the **path**
  - the algorithm ends after a polynomial number of iterations independent of the edge capacities