Algorithms for Data Processing
Lecture VI: Network Flows – Applications

Alessandro Artale

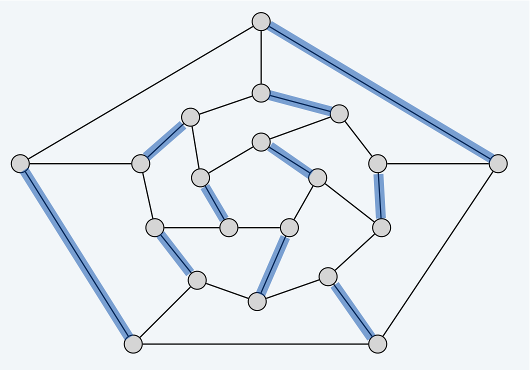
Free University of Bozen-Bolzano
Faculty of Computer Science
http://www.inf.unibz.it/˜artale
artale@inf.unibz.it

2019/20 – First Semester
MSc in Computational Data Science — UNIBZ

Some material (text, figures) displayed in these slides is courtesy of:
Alberto Montresor, Werner Nutt, Kevin Wayne, Jon Kleinberg, Eva Tardos.

Matching

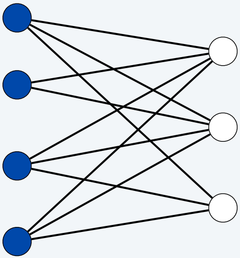
Matching. Given an undirected graph $G = (V, E)$ a subset of edges $M \subseteq E$ is a matching if each node appears in at most one edge in M.



Bipartite Graphs – 2-Colorability

Bipartite Graph: An undirected graph $G = (V, E)$ is bipartite if the vertices can be colored blue or white such that every edge has one white and one blue end.

- Applications.
  Matching: residents = blue, hospitals = white;
  Scheduling: machines = blue, jobs = white.

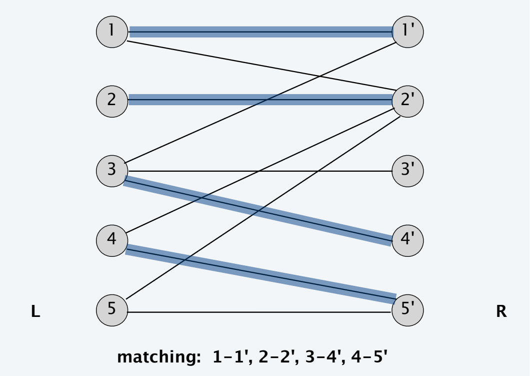


a bipartite graph

Bipartite Matching

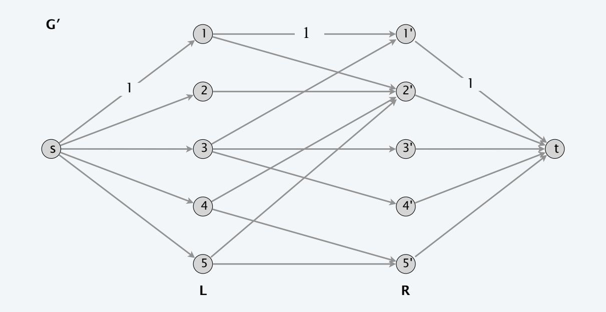
Bipartite Graph: A graph G is bipartite if the nodes can be partitioned into two subsets L and R such that every edge connects a node in L with a node in R.

Bipartite Matching. Given a bipartite graph G find a max-cardinality matching.



Bipartite Matching: Max-Flow based Algorithm

- Create a digraph $G' = (L \cup R \cup \{s,t\}, E')$;
- Direct all edges from L to R and assign unit capacity;
- Add unit-capacity edges from s to each node in L;
- Add unit-capacity edges from each node in R to t;
- The value of the maximum s-t flow in this network G' is equal to the size of the maximum matching in G.



Max-Flow based Algorithm: Proof of Correctness

- By the Integrality Theorem there is a max-flow f_m in G' of value $val(f_m) = k$;
- Since all capacities $= 1$, each $f(e)$ is equal to either 0 or 1;
- Let M' be the set of edges (x, y) such that $x \in L,\ y \in R$, and $f(x, y) = 1$;
- Prop/1. *M' contains $val(f_m) = k$ edges.*
  - Consider the cut $(S = L \cup \{s\}, T = R \cup \{t\})$, and apply the Flow value Lemma:
    $k = val(f_m) = \sum_{e\ out\ of\ S} f(e) - \sum_{e\ into\ S} f(e) = |M'| - 0$

Max-Flow based Algorithm: Proof of Correctness/2

- Prop/2. M' is a matching.
  1. Each node in L is the tail of at most one edge in M'.
     By contradiction, there would be a node $x \in L$ tail of two edges in M'. So, flow out of $x \geq 2$ which violates the conservation condition.
  2. Each node in R is the head of at most one edge in M'.
- Prop/3. M' has maximal size.
  - Let M_1 be a matching having edges $(x_1, y_1), \ldots, (x_p, y_p)$, with $p > k$;
  - Consider the flow f that sends one unit along each path of the form $s \rightarrow x_i \rightarrow y_i \rightarrow t$, for $i = 1, \ldots, p$;
  - We can easily show that f is an s-t flow of value $p > k$, which contradicts that k is the value of the max-flow.

Bipartite Matching: Run Time

- Let $n = |X| = |Y|$, and let m be the number of edges of G.
- Since $C = 1$, we can use the Ford-Fulkerson algorithm to find the max-flow.

Bipartite Matching: Run Time. The Ford-Fulkerson Algorithm can be used to find a maximum matching in a bipartite graph in $O(mn)$ time.

Perfect Matchings in Bipartite Graphs

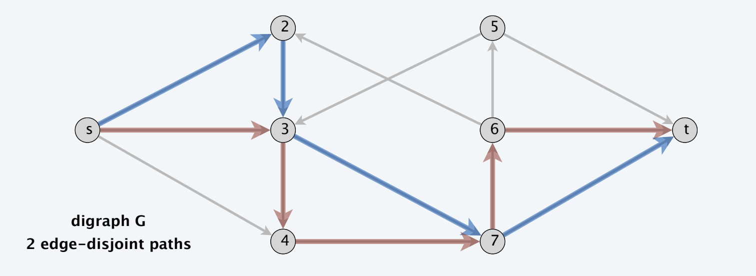
Definition. Given a bipartite graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in M.

Perfect Matching Algorithm. We use the algorithm for Bipartite Matching and then check if this matching is perfect.

Edge-Disjoint Paths

Definition. Given a graph $G = (V, E)$, two paths are edge-disjoint if they have no edge in common.

Definition [Directed Edge-Disjoint paths problem.] Given a *directed* graph $G = (V, E)$ and two distinguished vertices s, t, find the max number of edge-disjoint $s \to t$ paths.



Edge-Disjoint Paths – Algorithm

Algorithm: Max-Flow formulation. Assign unit capacity to every edge and show that the max-flow solves the problem, i.e., the max number of edge-disjoint s-t paths = value of max flow.

We show the correctness of this Algorithm by showing the following two Lemmas.

Lemma 1. If there are k edge-disjoint s-t paths in a directed graph G, then the value of the maximum s-t flow in G is at least k.
Proof.

- Set $f(e) = 1$ if e participates in some path P_j; else set $f(e) = 0$;
- Since paths are edge-disjoint, then f is a flow, and $val(f) = k$.

Edge-Disjoint Paths – Algorithm/2

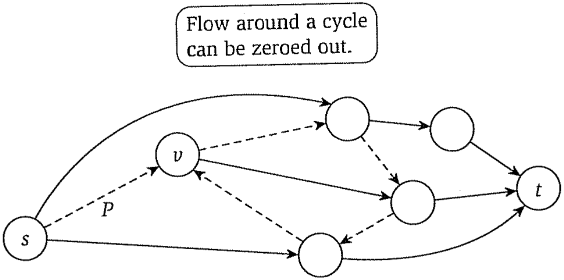
We now show that also the converse holds.

Lemma 2. If f is a 0-1 valued flow with $val(f) = k$, then the set of edges with $f(e) = 1$ contains a set of k edge-disjoint paths.

Proof by induction on the number, m, of edges with $f(e) = 1$.

- [Base Case.] $m = 0$. Then $k = 0$, and there is no path. Thus the Lemma holds.
- [Inductive Step.] $m \geq 1$. Then, $k \geq 1$. Let (s, u) with $f(s, u) = 1$, by flow conservation, there exists an edge (u, v) with $f(u, v) = 1$. Continue until we reach either t or an already visited node, v.
- [Case 1.] We found an $s \to t$ path, P. Consider a new flow, f', obtained by decreasing the flow values on the edges along P to 0. Then, $val(f') = k - 1$ and there are $m' < m$ edges carrying a flow. By IH, we get $k - 1$ disjoint paths associated to the flow f' and adding P we obtain k disjoint paths.

Edge-Disjoint Paths – Algorithm/3



- [Case 2.] Consider the cicle C involving node v. Consider a new flow, f', obtained by decreasing the flow values on the edges along C to 0. Then, $val(f') = k$ and there are $m' < m$ edges carrying a flow. By IH, we get k disjoint paths associated to the flow f'.

Thus the Lemma holds!

Edge-Disjoint Paths – Algorithm/4

We proved the following:

Thorem 1. There are k edge-disjoint paths in a directed graph G from s to t if and only if the value of the maximum value of an s-t flow in G is at least k.

Path Extraction. The proof of Lemma 2 provides a procedure for constructing the k paths, given a max flow in G. This procedure is sometimes referred to as a path decomposition of the flow.

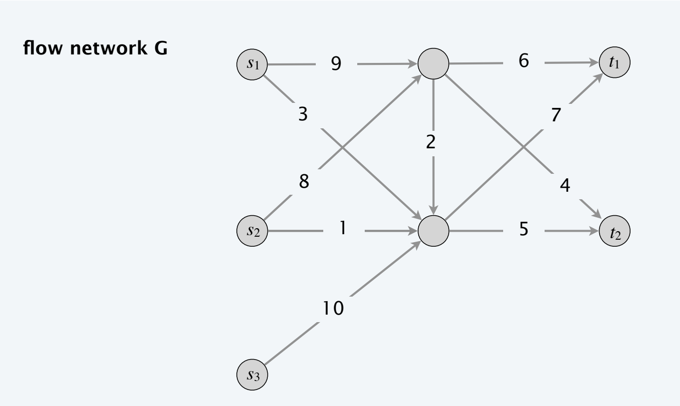
Run Time Analysis. The algorithm as provided in the Proof of Lemma 2 runs in $O(mn)$.

Generalisation of the Max-Flow Problem

- Multiple sources and multiple sinks.
- New conservation conditions.
- Lower bounds on edge flows.

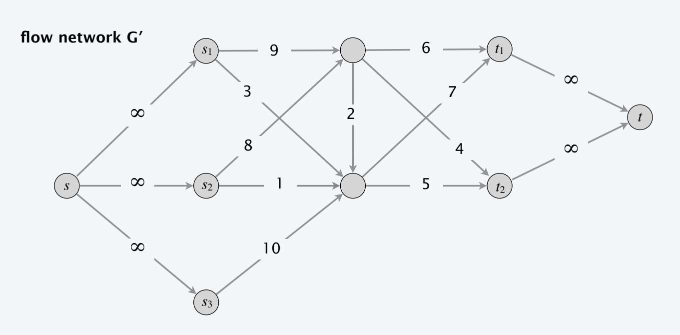
Multiple Sources and Sinks

Definition. Given a directed graph $G = (V, E)$ with edge capacities $c(e) > 0$ and multiple source nodes and multiple sink nodes, find a max flow that can be sent from the source nodes to the sink nodes.



Multiple Sources and Sinks: Max-Flow Formulation

- Add a new source node s and sink node t;
- For each original source node s_i add edge (s, s_i) with capacity ∞;
- For each original sink node t_j, add edge (t_j, t) with capacity ∞.



Circulation with Supplies and Demands

Definition. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$, we associate to each node a demand, $d(v) \in \mathbb{Z}$, such that:

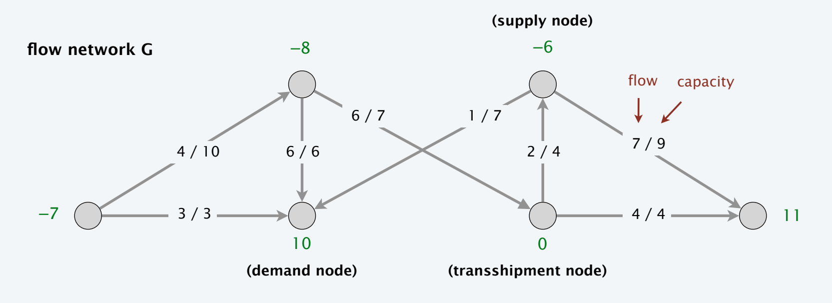
- $d(v) < 0$. The node is a supply node: the node is a source wishing to send out $-d(v)$ units more flow than it receives.
- $d(v) > 0$. The node is a demand node: the node is a sink wishing to receive $d(v)$ units more flow than it sends.

Circulation with Supplies and Demands/2

Definition. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$ and demand $d(v) \in \mathbb{Z}$, a circulation is a function $f(e)$ that satisfies:

[Capacity Condition.] For each node $e \in E$: $0 \leq f(e) \leq c(e)$;

[Demand Condition.] For each vertex $v \in V$: $\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$.



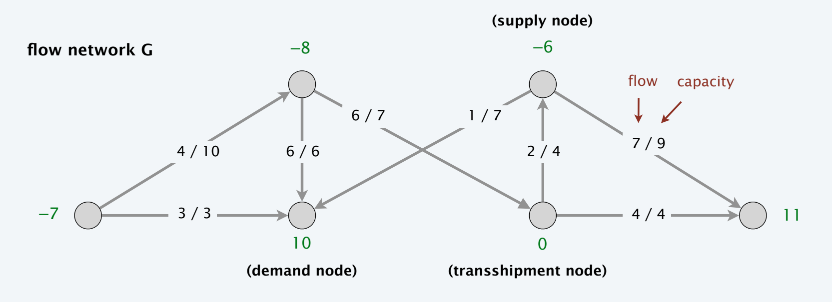
Feasible Circulations

Feasible Circulation Problem. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$ and demand $d(v) \in \mathbb{Z}$, check whether there exists a circulation that meets both *capacity* and *demand* conditions.

Feasible Circulations

Feasible Circulation Problem. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$ and demand $d(v) \in \mathbb{Z}$, check whether there exists a circulation that meets both *capacity* and *demand* conditions.

The flow value in the following graph represents a feasible circulation.



Feasible Circulations/2

There is a simple conservation condition that must hold in order for a feasible circulation to exist:

Property. If there exists a feasible circulation with demand $d(v)$, then $\sum_v d(v) = 0$, i.e.,

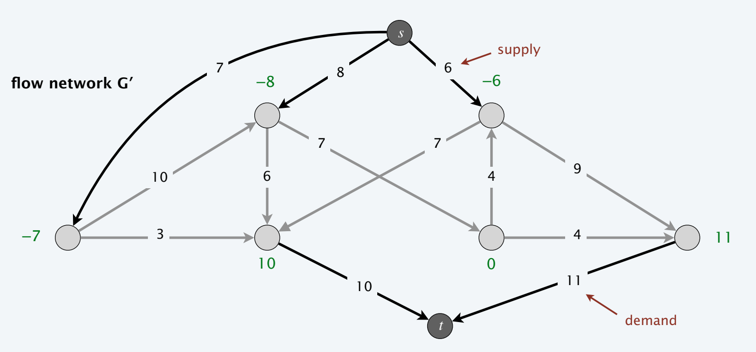
$$D = \sum_{d(v)>0} d(v) = \sum_{d(v)<0} -d(v)$$

.

Feasible Circulations solved with Max-Flow

Starting from G generate a graph G' as follows:

- Add new source s and sink t;
- For each vertex v with $d(v) < 0$, add edge (s, v) with capacity $-d(v)$;
- For each vertex v with $d(v) > 0$, add edge (v, t) with capacity $d(v)$.



Feasible Circulations solved with Max-Flow/2

Theorem. There is a feasible circulation with demands $d(v)$ in G if and only if the maximum s-t flow in G' has value D.

Proof.

- There cannot be an s-t flow in G' of value greater than D since the cut (A, B) with $A = \{s\}$ has $c(A, B) = D$.
- $(\Rightarrow)$ If there is a feasible circulation f with demands $d(v)$ in G, then by sending a flow value of $-d(v)$ on each edge (s, v), and a flow value of $d(v)$ on each edge (u, t), we obtain an s-t flow in G' of value D, and by the min-cut/max-flow Theorem this is a max-flow.
- $(\Leftarrow)$ Conversely, suppose there is a max s-t flow in G' of value D.
  - Then, each edge out of s, and each edge into t, is completely saturated with flow;
  - If we delete these edges, we obtain a circulation f in G with $\sum_{e \ into \ v} f(e) - \sum_{e \ out \ of \ v} f(e) = d(v)$, for every $v \in V$.

Circulation with Supplies, Demands, and Lower Bounds

In many applications, we want to force the flow to make use of certain edges. This can be enforced by placing lower bounds on edges.

Definition. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$, lower bounds $\ell(e) \geq 0$ and demand $d(v) \in \mathbb{Z}$, a circulation is a function $f(e)$ that satisfies:

[Capacity Condition.] For each node $e \in E$: $\ell(e) \leq f(e) \leq c(e)$;

[Demand Condition.] For each vertex $v \in V$: $\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$.

Circulation with Supplies, Demands, and Lower Bounds

In many applications, we want to force the flow to make use of certain edges. This can be enforced by placing lower bounds on edges.

Definition. Given a directed graph $G = (V, E)$, with edge capacities $c(e) > 0$, lower bounds $\ell(e) \geq 0$ and demand $d(v) \in \mathbb{Z}$, a circulation is a function $f(e)$ that satisfies:

[Capacity Condition.] For each node $e \in E$: $\ell(e) \leq f(e) \leq c(e)$;

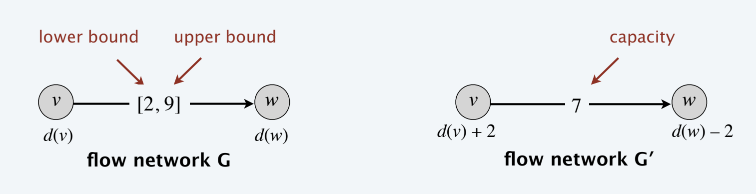
[Demand Condition.] For each vertex $v \in V$: $\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$.

Circulation problem with lower bounds. Given (V, E, ℓ, c, d), does there exist a feasible circulation?

Circulation with Supplies, Demands, and Lower Bounds/2

Max-flow formulation. Model lower bounds as circulation with demands (but no lower bounds)

- Start with a flow f_0 s.t. on every edge in G, $f_0(e) = \ell(e)$, to satisfy the lower bounds.
- Add a new flow f' s.t. $f_0 + f'$ is a feasible circulation in G, when f' is a feasible circulation in G' without lower bounds with demands d' and capacity c':
  - For each $v \in V$, $(f_0^{in}(v) - f_0^{out}(v)) + (f'^{in}(v) - f'^{out}(v)) = d(v)$, i.e.,
    $d'(v) = d(v) - (f_0^{in}(v) - f_0^{out}(v)) = d(v) - \Big(\sum_{e \text{ into } v} \ell(e) - \sum_{e \text{ out of } v} \ell(e) \Big)$
  - For each $e \in E$, $c'(e) = c(e) - \ell(e)$



Circulation with Supplies, Demands, and Lower Bounds/3

Theorem. There exists a circulation in G iff there exists a circulation in G'.

Proof Sketch. f is a circulation in G iff there exists a circulation f' in G' s.t. $f(e) = f'(e) + f_0(e)$.

Survey Design

- We consider here a task faced by many companies wanting to measure customer satisfaction;
- We illustrates how the Bipartite Matching Problem is useful to balance decisions across a set of options:
    - designing questionnaires by balancing relevant questions across a population of consumers.
- A major issue in the field of data mining to study consumer preference patterns.
    - A company wishing to conduct a survey, sending customized questionnaires to a particular group of n customers to determine which products people like.

Survey Design Guidelines

- A customer can only be asked about products that he or she has purchased (think about "Shopper Cards");
- Ask consumer i between c_i and c'_i number of products;
- Ask between p_j and p'_j distinct consumers about a given product j.

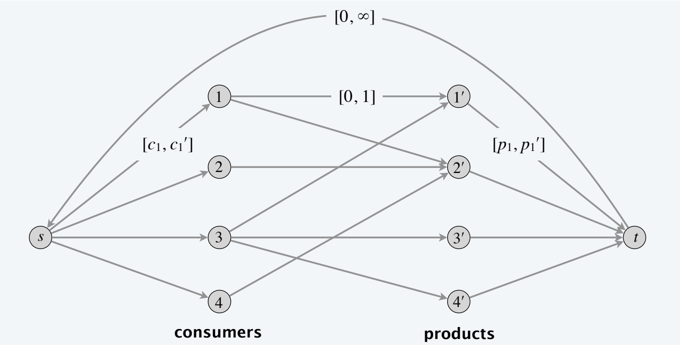
Problem. Decide if there is a way to design a questionnaire for each customer so as to satisfy all these conditions.

Survey Design as a Bipartite Matching Problem

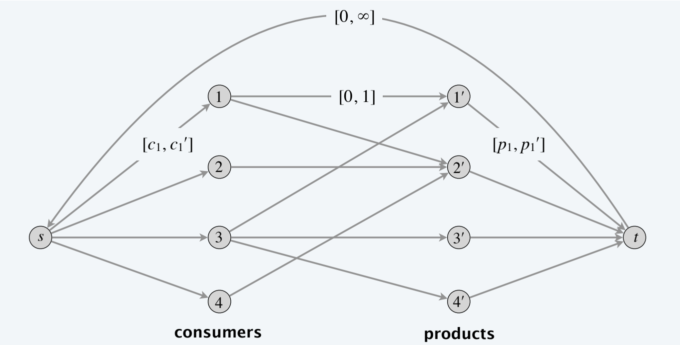
Max-flow formulation. Model as a Bipartite Matching Problem together with a circulation problem with lower bounds.

- Nodes are the customers and the products;
- Add edge (i, j) if customer i purchased product j;
- Add edges from s to customer i, from product j to t;
- Demands are all set to 0;
- Let $e = (s, i)$, then $c(e) = [c_i, c'_i]$;
- Let $e = (j, t)$, then $c(e) = [p_j, p'_j]$;
- Let $e = (i, j)$, then $c(e) = [0, 1]$.

Survey Design as a Bipartite Matching Problem/2



Survey Design as a Bipartite Matching Problem/2



Theorem. The max-flow formulation of a survey design has a feasible circulation if and only if there is a feasible (i.e., respecting all the guidelines) way to design the survey.

Airline Scheduling

Airline scheduling problem.

- Complex computational problem faced by airline carriers;
- Must produce large number of schedules that are efficient in terms of equipment usage, crew allocation, and customer satisfaction;
- Deal with unpredictable issues like weather and breakdowns;
- One of the largest consumers of high-powered algorithmic techniques.

Airline Scheduling

Airline scheduling problem.

- Complex computational problem faced by airline carriers;
- Must produce large number of schedules that are efficient in terms of equipment usage, crew allocation, and customer satisfaction;
- Deal with unpredictable issues like weather and breakdowns;
- One of the largest consumers of high-powered algorithmic techniques.

We concentrate on the resource allocation problem.

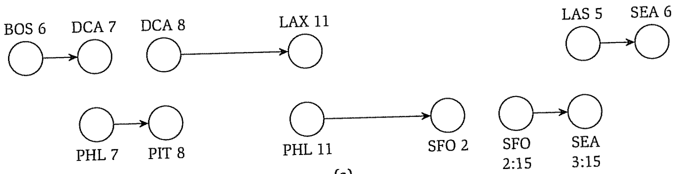
- Input: set of m flight routes for a given day.
- Each flight route i has: origin o_i, starting time s_i and arrives at destination d_i at final time f_i.

Airline Scheduling/2

Goal in this problem. Determine whether it is possible to serve all m flight routes on your original list, using at most k planes in total.

In order to do this, you need to find a way of efficiently reusing planes for multiple flights routes.

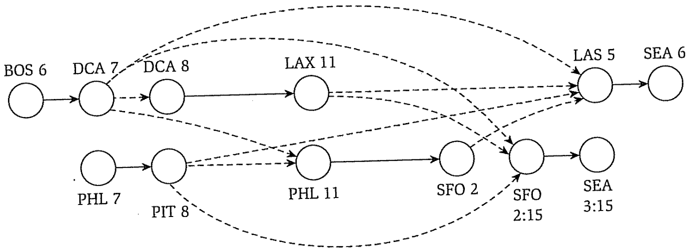
Airline Scheduling Example



(a)

1. Boston (depart 6am) – Washington DC (arrive 7am)
2. Philadelphia (depart 7am) – Pittsburgh (arrive 8am)
3. Washington DC (depart 8am) – Los Angeles (arrive 11am)
4. Philadelphia (depart 11am) – San Francisco (arrive 2pm)
5. San Francisco (depart 2:15pm) – Seattle (arrive 3:15pm)
6. Las Vegas (depart 5pm) – Seattle (arrive 6pm)

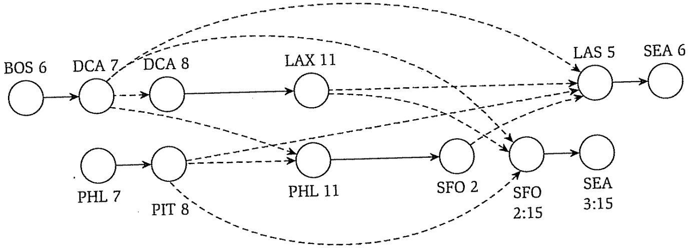
Airline Scheduling Example/2



Reachable Flight Routes. Whenever the same plane can be reused for different flight routes, for example according to these rules:

1. The destination of flight route i is the same as the origin of j, and there is enough time to perform maintenance on the plane in between; or
2. A flight route can be added in between that gets the plane from the destination of i to the origin of j with adequate time in between.

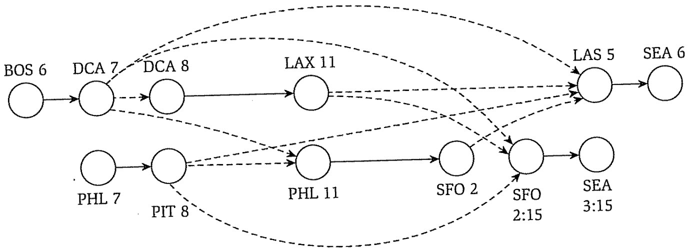
Airline Scheduling Example/3



Solution with 2 planes.

- Plane 1: (1), (3), (6) is not a solution:
  - Not enough maintenance time in San Francisco between flights (4) and (5).

Airline Scheduling Example/3



Solution with 2 planes.

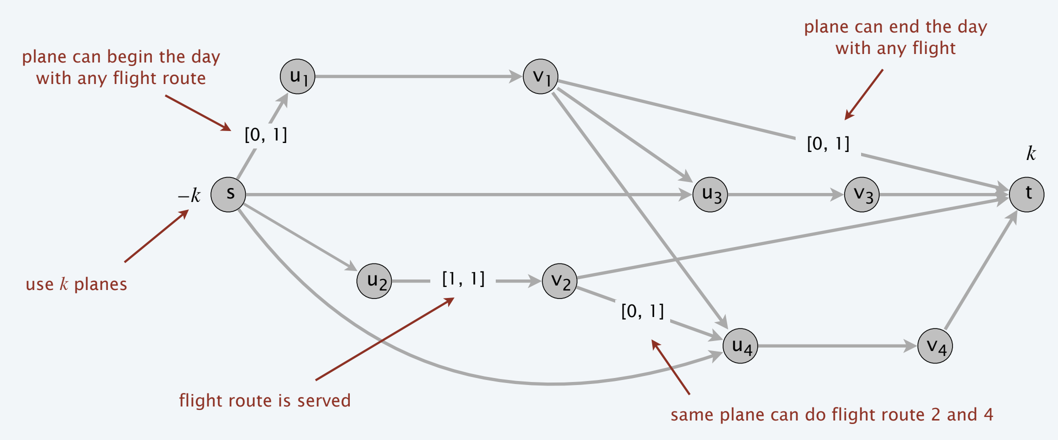
- Plane 1: (1), (3), (6) is not a solution:
  - Not enough maintenance time in San Francisco between flights (4) and (5).
- Plane 1: (1), (3), (5)
- Plane 2: (2), (4), (6)

Airline Scheduling – Net Flow Solution

Circulation formulation: To see if k planes suffice we construct the following circulation graph G.

- Units of flow will correspond to planes;
- There is an edge (u_i, v_i) for each flight route i with upper and lower capacity bounds of 1 to enforce that exactly one unit of flow (i.e., plane) serves the flight route;
- If flight route j is reachable from flight route i add edge (v_i, u_j) with capacity 1;
- Add source node s with edges (s, u_i) and capacity 1 (*a plane can begin the day with any flight route*);
- Add sink t with edges (v_i, t) with capacity 1 (*a plane can finish the day with any flight route*).
- the node s will have a demand of -k, and the node t will have a demand of k. All other nodes will have a demand of 0.

Airline Scheduling – Net Flow Solution/2



Airline Scheduling – Algorithm Analysis

Theorem. There is a way to serve all flight routes using k planes if and only if there is a feasible circulation in the circulation graph G.

Note: To output the flight routes assigned to a given plane is enough to generate the paths with edge (s, u_i) that carries one unit of flow (the problem is similar to the edge-disjoint paths).

Airline Scheduling – Algorithm Analysis

Theorem. There is a way to serve all flight routes using k planes if and only if there is a feasible circulation in the circulation graph G.

Note: To output the flight routes assigned to a given plane is enough to generate the paths with edge (s, u_i) that carries one unit of flow (the problem is similar to the edge-disjoint paths).

How do you modify the algorithm to allow at most k planes?

Thank You!