Applications of Maximum Flows and Minimum Cuts

SURVEY DESIGN

SURVEY DESIGN

- **The Problem**

- Consider a company that sells k products and has a database containing the purchase histories of a large number of customers.

- The company wishes to conduct a survey, sending customized questionnaires to a particular group of n of its customers, to try determining which products people like overall.

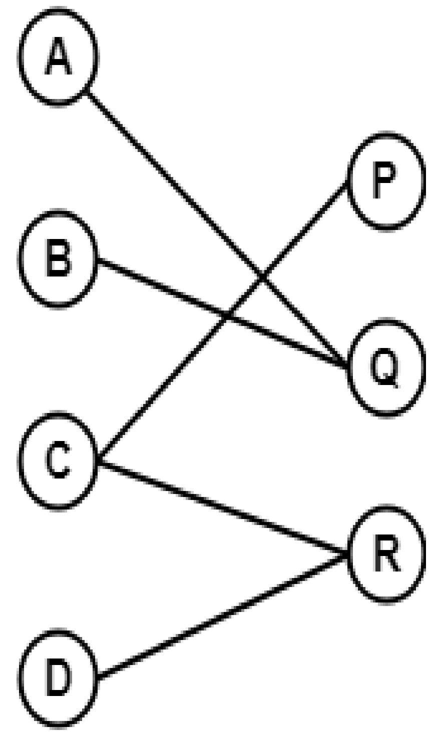
SURVEY DESIGN

- **Guidelines for Survey Design**
- Each customer will receive questions about a certain subset of the products.
- A customer can only be asked about products that he or she has purchased.
- To make each questionnaire informative, but not too long so as to discourage participation, each customer i should be asked about a number of products between ci and ci'.
- Finally, to collect sufficient data about each product, there must be between pj and pj' distinct customers asked about each product j.

SURVEY DESIGN

- The input to the survey problem consists of a **bipartite graph G** whose nodes are the customers and the products.

- There is an edge between customer i and product j, if he or she has ever purchased product j.



Bipartite Graph

SURVEY DESIGN

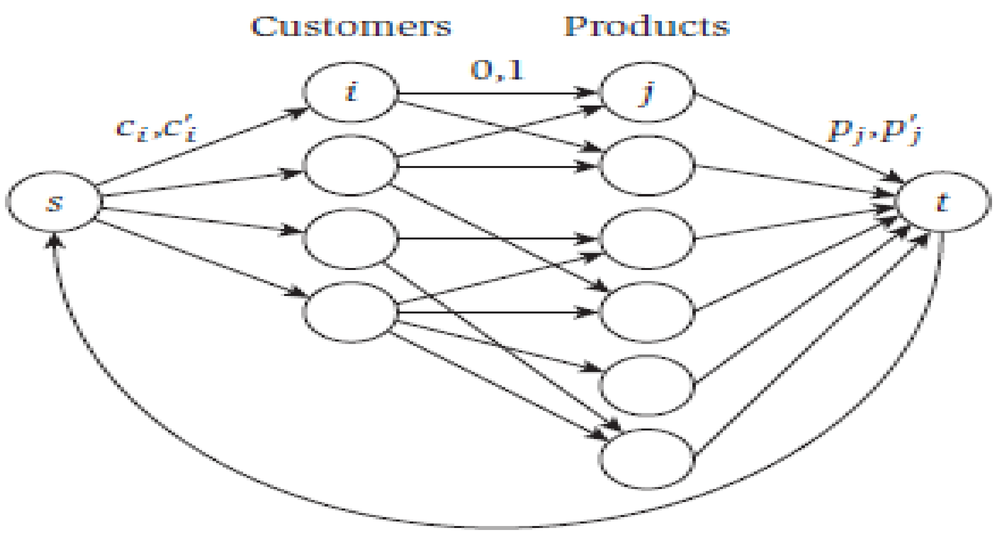
- For each customer $i = 1\ldots\ldots n$, we have limits $\mathbf{c_i \leq c_i'}$ on the number of products he or she can be asked about.

- For each product $j = 1\ldots\ldots\ldots k$ we have limits $p_j \leq p_j'$ j on the number of distinct customers that have to be asked it.

- The problem is to decide if there is a way to design a questionnaire for each customer so as to satisfy all these conditions.

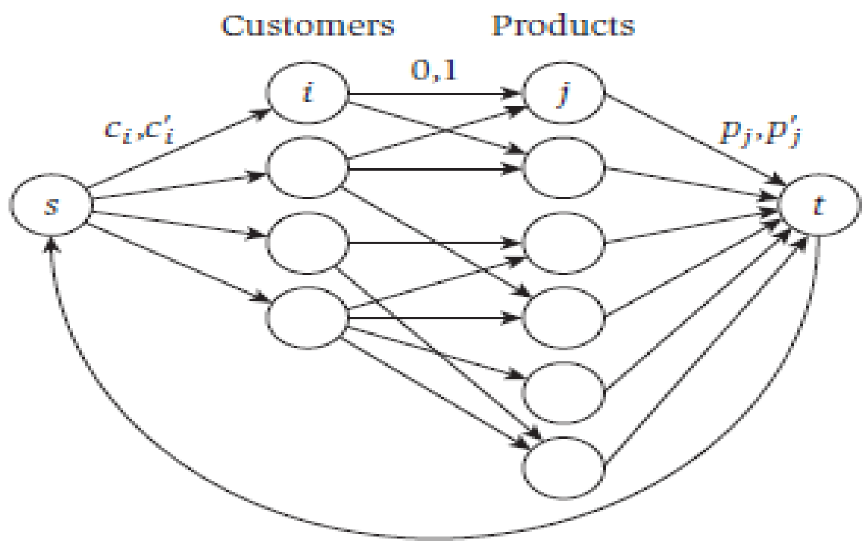
DESIGNING THE ALGORITHM

- To obtain the graph G' from G, we orient the edges of G from customers to products, add nodes s and t with edges (s, i) for each customer i = 1, . . . , n, edges (j, t) for each product j = 1, . . . , k, and an edge (t, s).



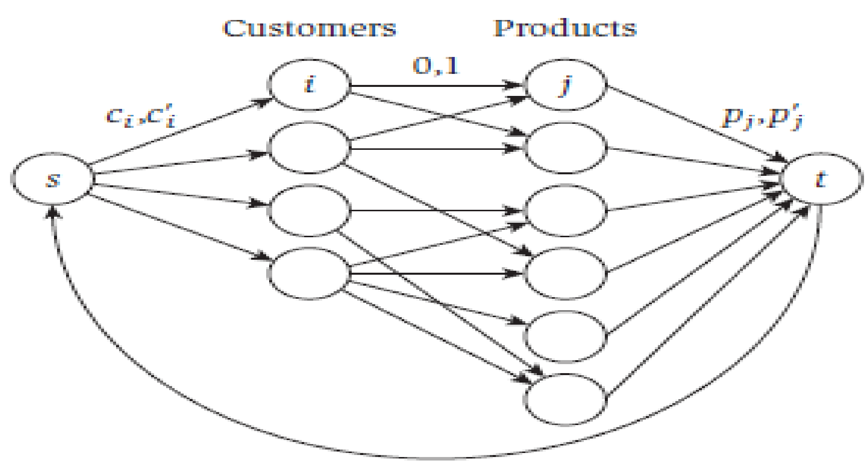
DESIGNING THE ALGORITHM

- The flow on the edge (s, i) is the number of products included on the questionnaire for customer i, so this edge will have a capacity of ci’ and a lower bound of ci.
- The flow on the edge (j, t) will correspond to the number of customers who were asked about product j, so this edge will have a capacity of pj’ and a lower bound of pj.



DESIGNING THE ALGORITHM

- Each edge (i, j) going from a customer to a product he or she bought has capacity 1, and 0 as the lower bound.
- The flow carried by the edge (t, s) corresponds to the overall number of questions asked.
- We can give this edge a capacity of $\sum_i c_i'$ and a lower bound of $\sum_i c_i$.



ANALYZING THE ALGORITHM

- **The graph G' with 0 demand, and the capacities and lower bounds given, has a feasible circulation if and only if there is a feasible way to design the survey.**
- **Proof:** The edge (i, j) will carry 1 unit of flow if customer "i" is asked about product "j" in the survey, and will carry no flow otherwise.
- The flow on the edges (s, i) is the number of questions asked from customer "i".
- The flow on the edge (j, t) is the number of customers who were asked about product "j".

ANALYZING THE ALGORITHM

- Finally, the flow on edge (t, s) is the overall number of questions asked.
- This flow satisfies the 0 demand, that is, there is flow conservation at every node.
- If the survey satisfies the rules, than the corresponding flow satisfies the capacities and lower bounds.
- Customer “i” will be surveyed about product “j” if and only if the edge (i, j) carries a unit of flow.

AIRLINE SCHEDULING

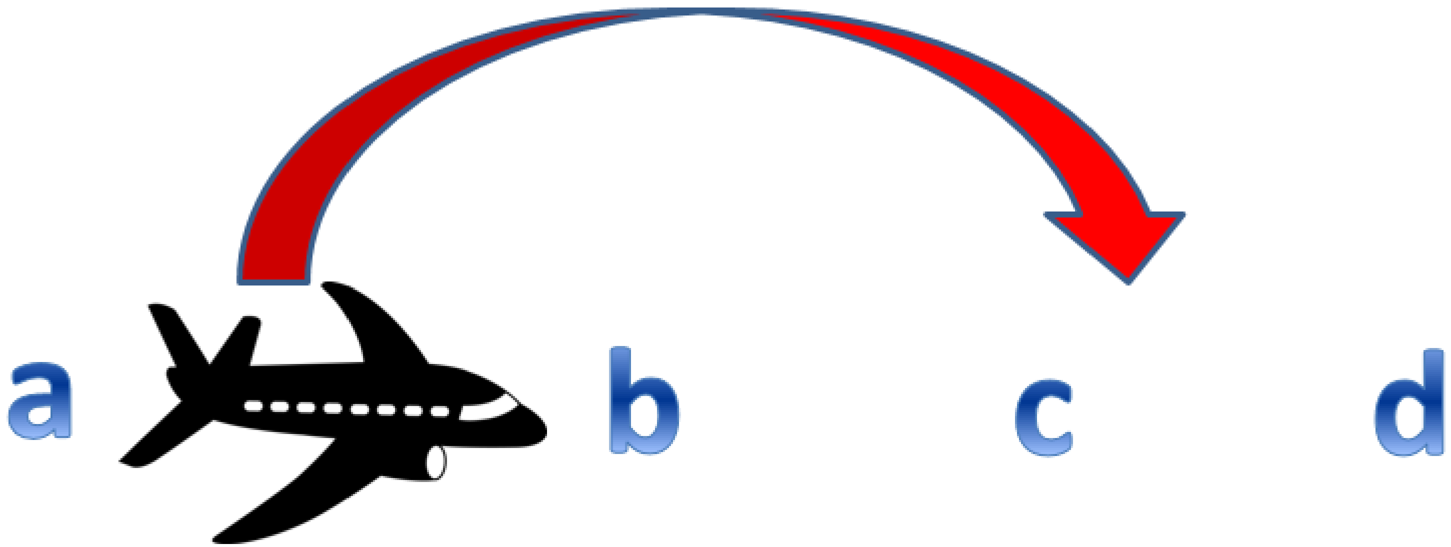
THE PROBLEM

- Suppose you’re in charge of managing a fleet of airplanes and you’d like to create a flight schedule for them.

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

THE PROBLEM

- Is it possible to use a single plane for a flight segment i, and then later for a flight segment j?



Airline Scheduling

- Is it possible to use a single plane for a flight segment i, and then later for a flight segment j?

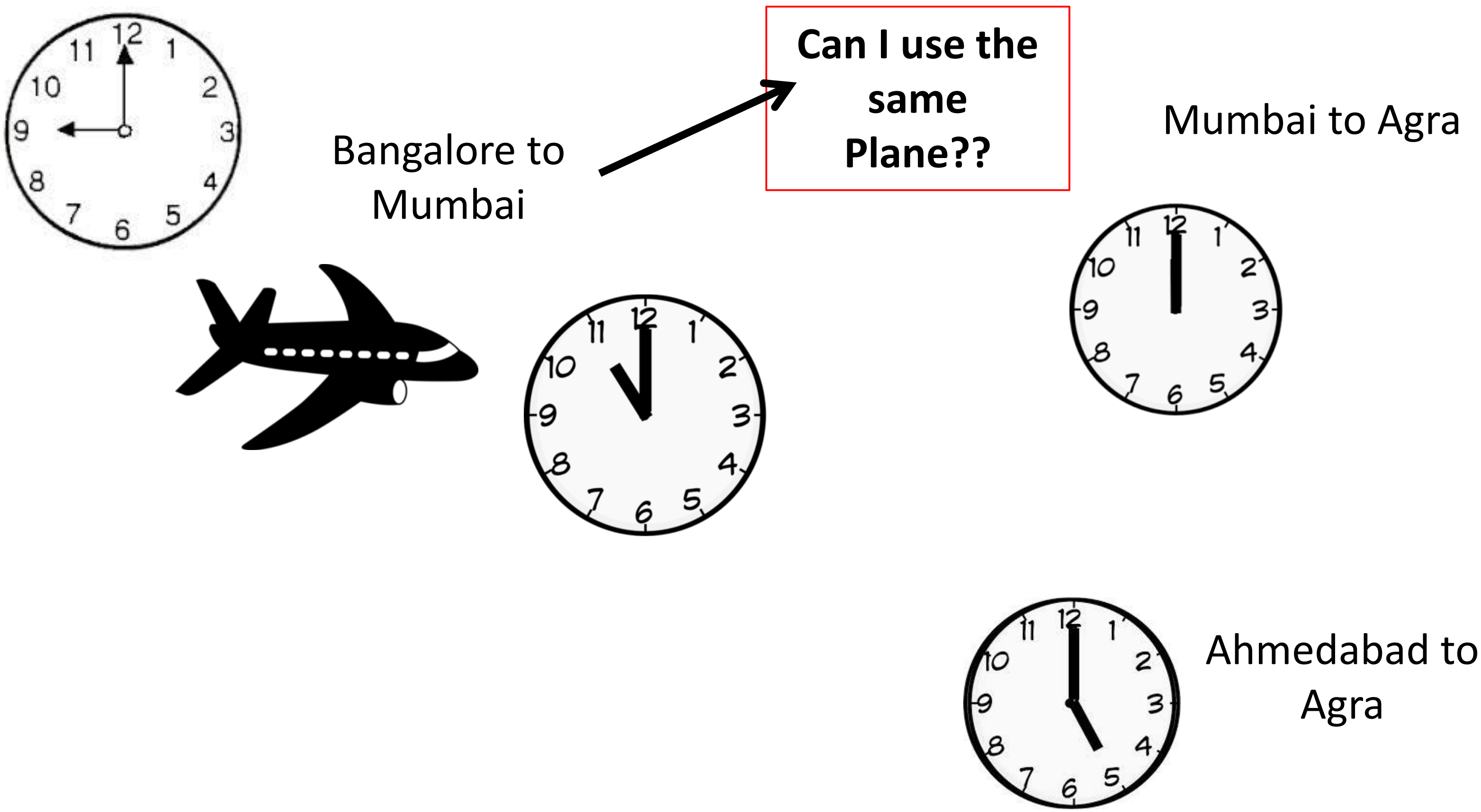
YES

- (a) the destination of i is the same as the origin of j, and there's enough time to perform maintenance on the plane in between;

or

- (b) you can add a flight segment in between that gets the plane from the destination of i to the origin of j with adequate time in between.

Airline Scheduling



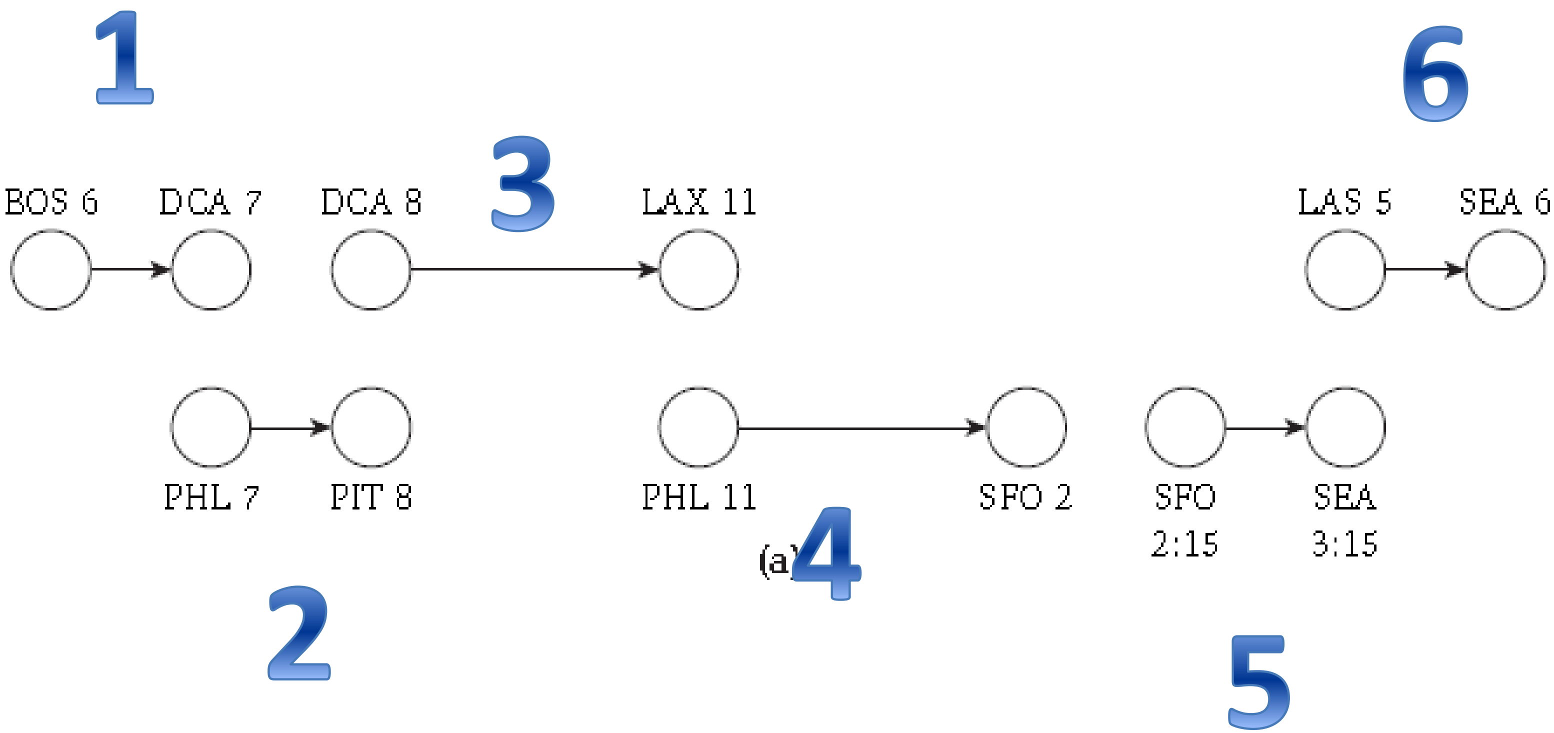
Airline Scheduling

- Is it possible to use a single plane for a flight segment i, and then later for a flight segment j?

YES

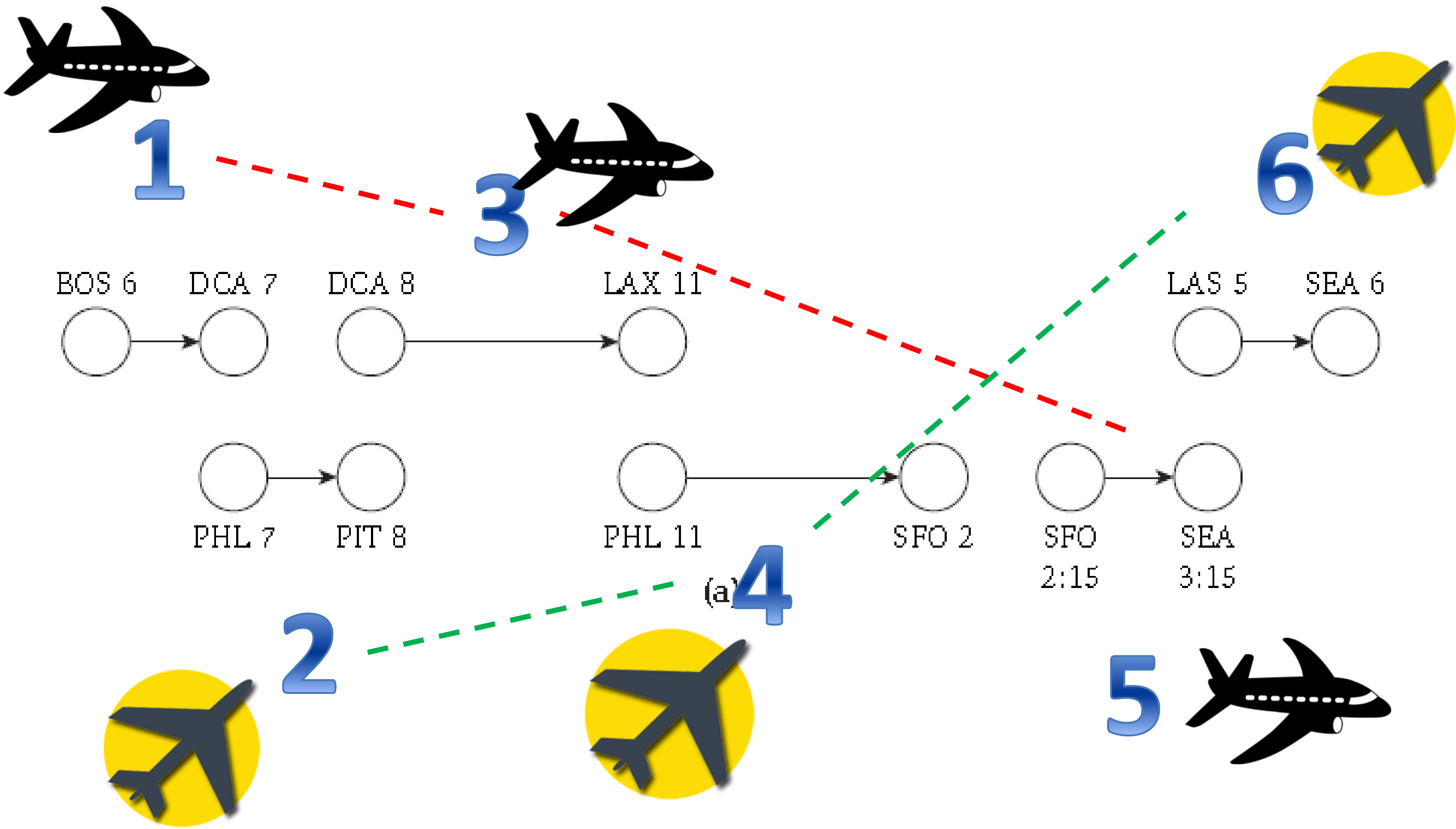
- **Goal**: Optimal number of planes needed for the given flight segments.

Airline Scheduling - Example



How many planes are needed to satisfy this fleet segment?

Airline Scheduling - Example



How many planes are needed to satisfy this fleet segment?

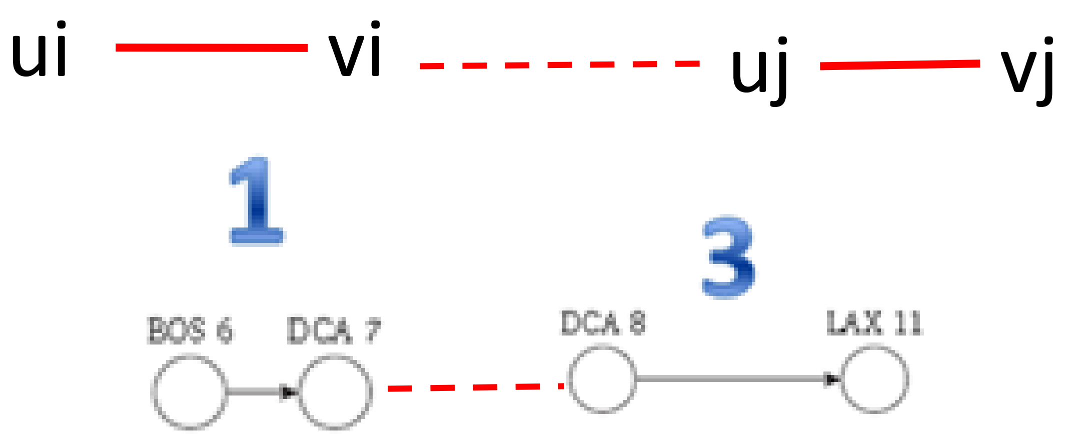
Designing the Algorithm

- We will have an edge for each flight, and upper and lower capacity bounds of 1 on these edges to require that exactly one unit of flow crosses this edge.

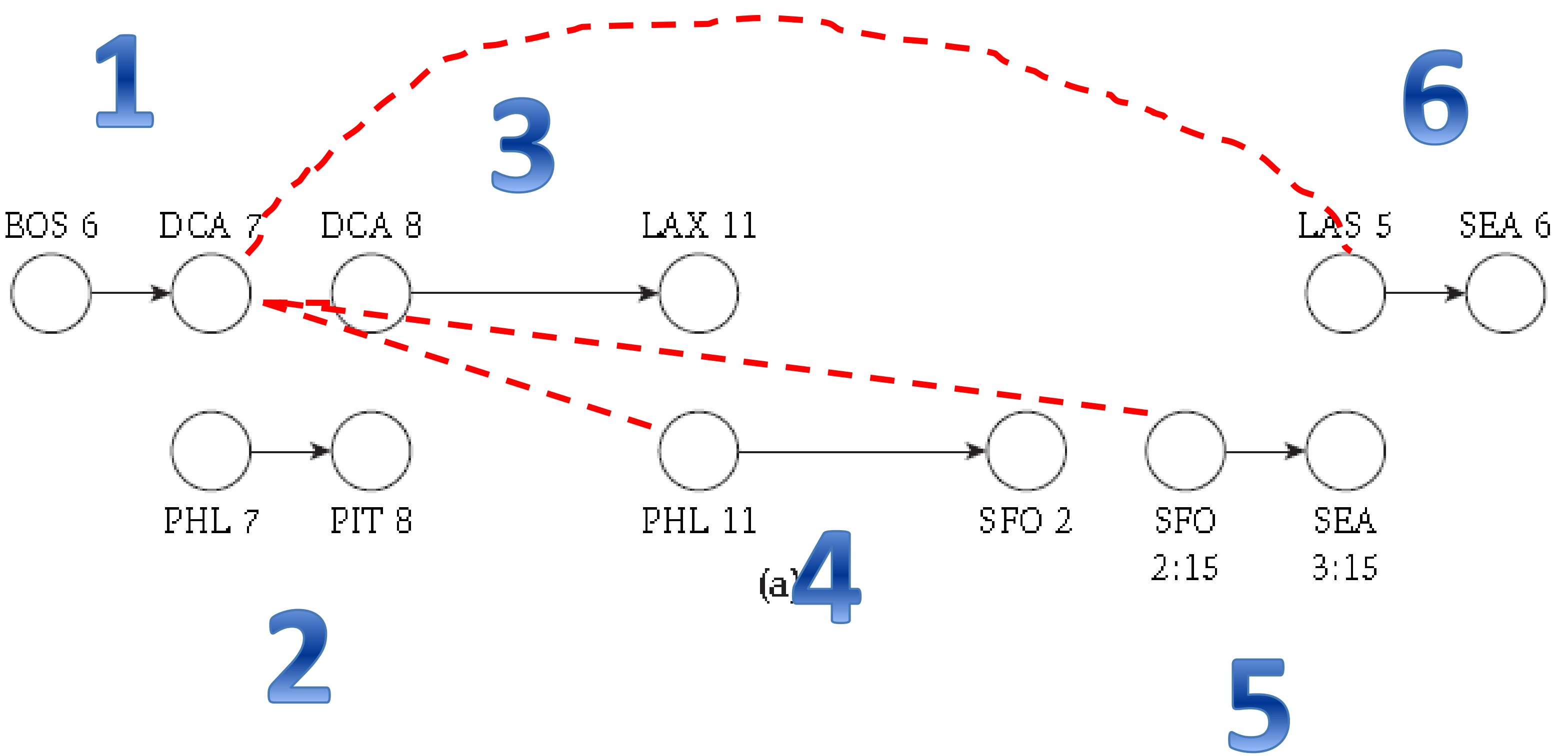
- In other words, each flight must be served by one of the planes.

Designing the Algorithm

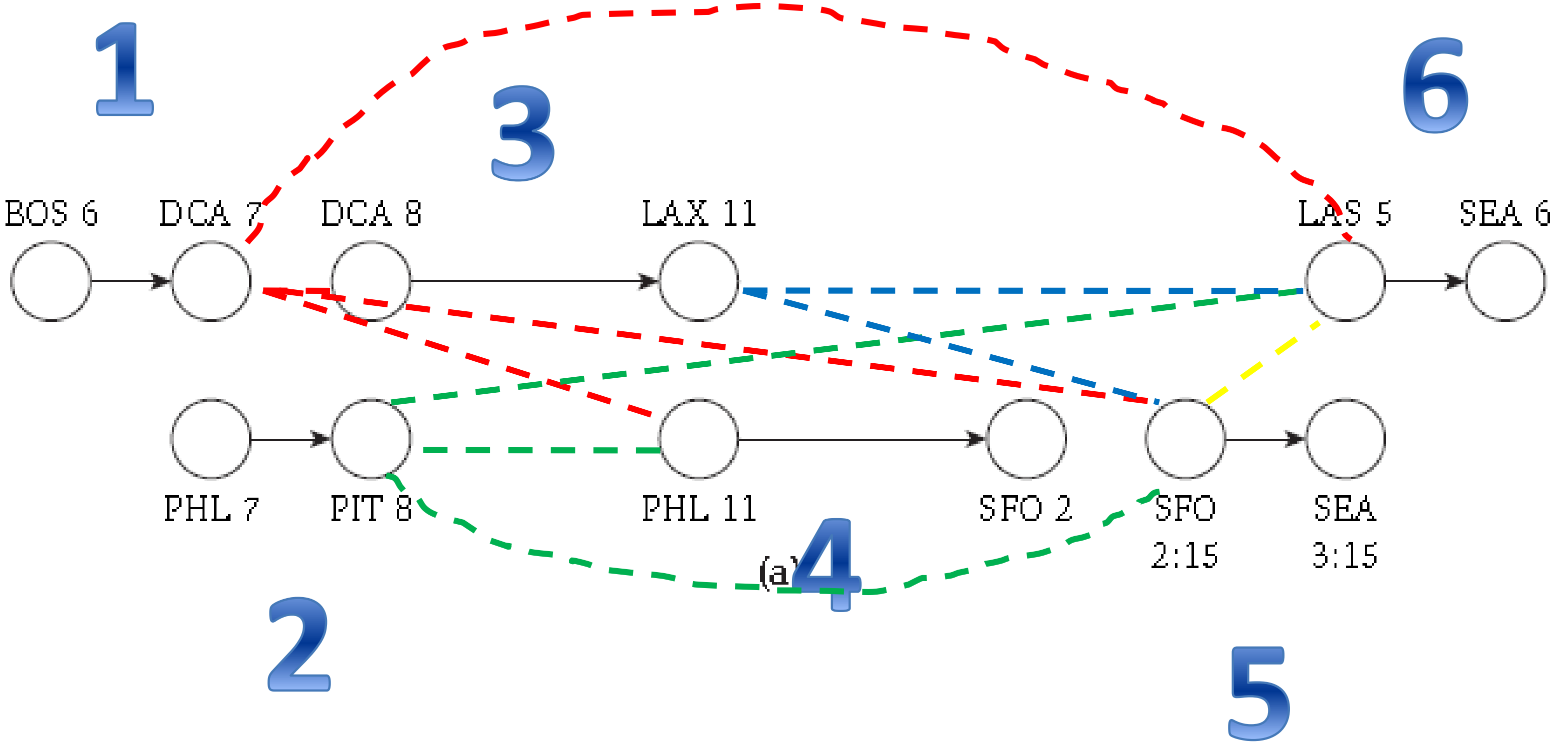
- If (ui , vi) is the edge representing flight i, and (uj , vj) is the edge representing flight j, and flight j is reachable from flight i, then we will have an edge from vi to uj with capacity 1.



Designing the Algorithm

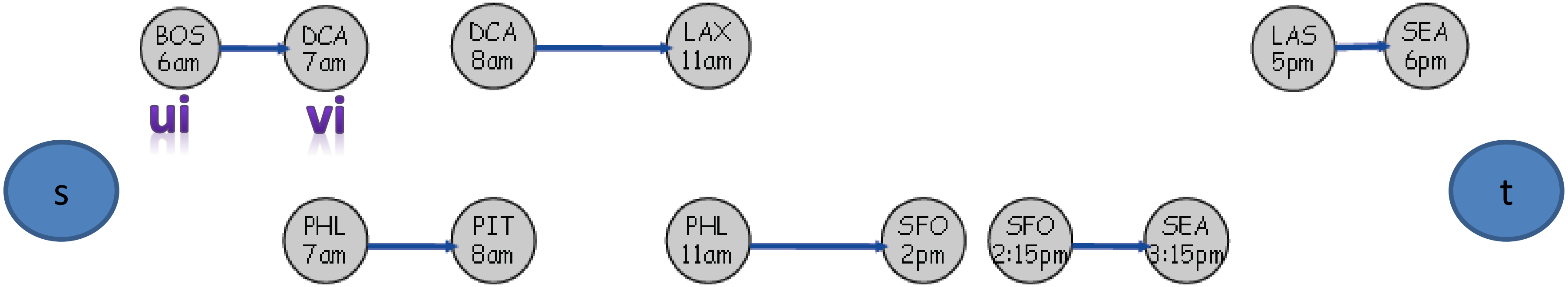


Designing the Algorithm - Example



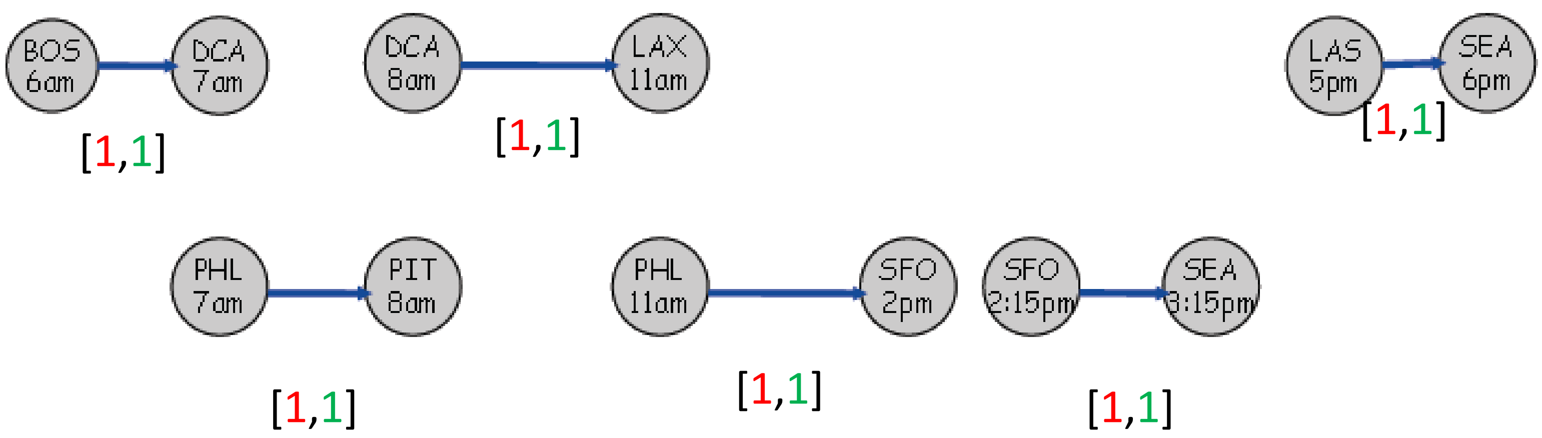
Designing the Algorithm

- The node set of the underlying graph G is defined as follows.
  - For each flight i, the graph G will have the **two nodes ui and vi.**
  - G will also have a **distinct source node s and sink node t.**



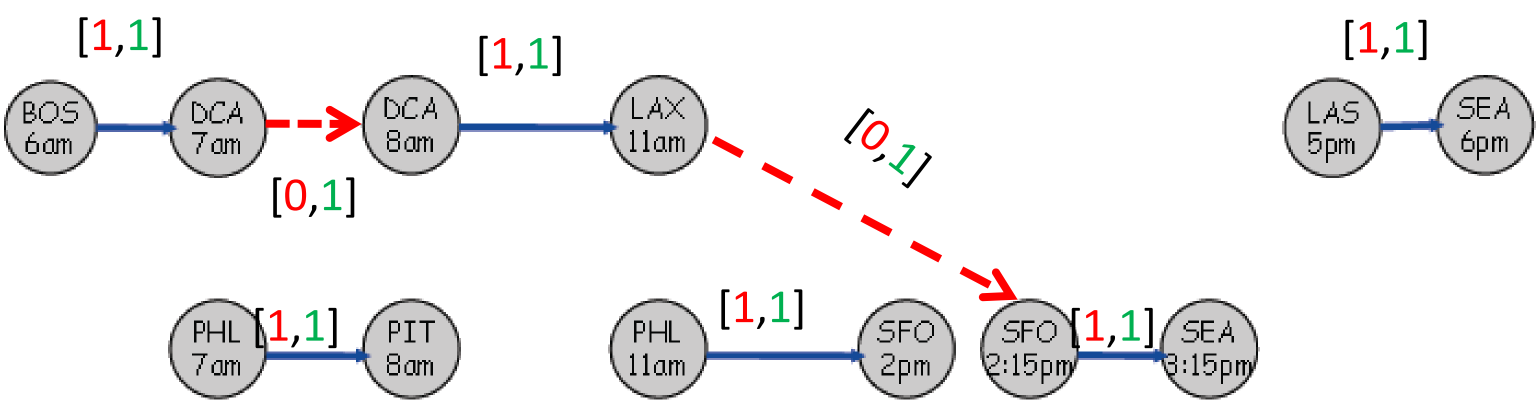
Designing the Algorithm

- The edge set of G is defined as follows.
  - For each i, **there is an edge (ui , vi)** with a lower bound of 1 and a capacity of 1. (Each flight on the list must be served.)



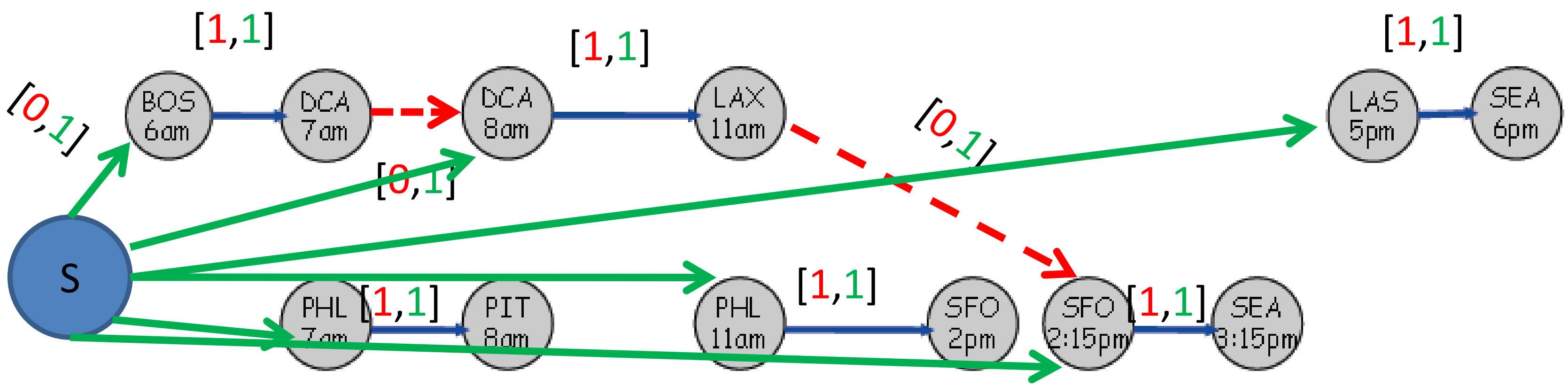
Designing the Algorithm

- The edge set of G is defined as follows.
  - For each i and j so that **flight j is reachable from flight i**, there is an edge **(vi , uj)** with a lower bound of 0 and a capacity of 1. (The same plane can perform flights i and j.)



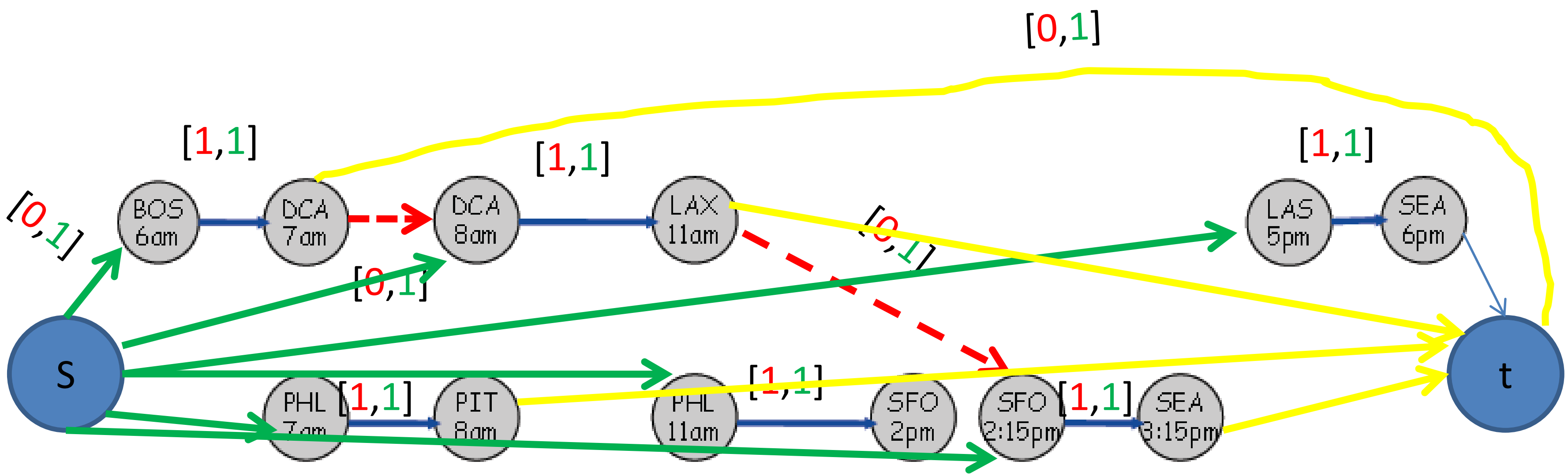
Designing the Algorithm

- The edge set of G is defined as follows.
  – For each i, there is an **edge (s, ui)** with a lower **bound of 0 and a capacity of 1**. (Any plane can begin the day with flight i.)



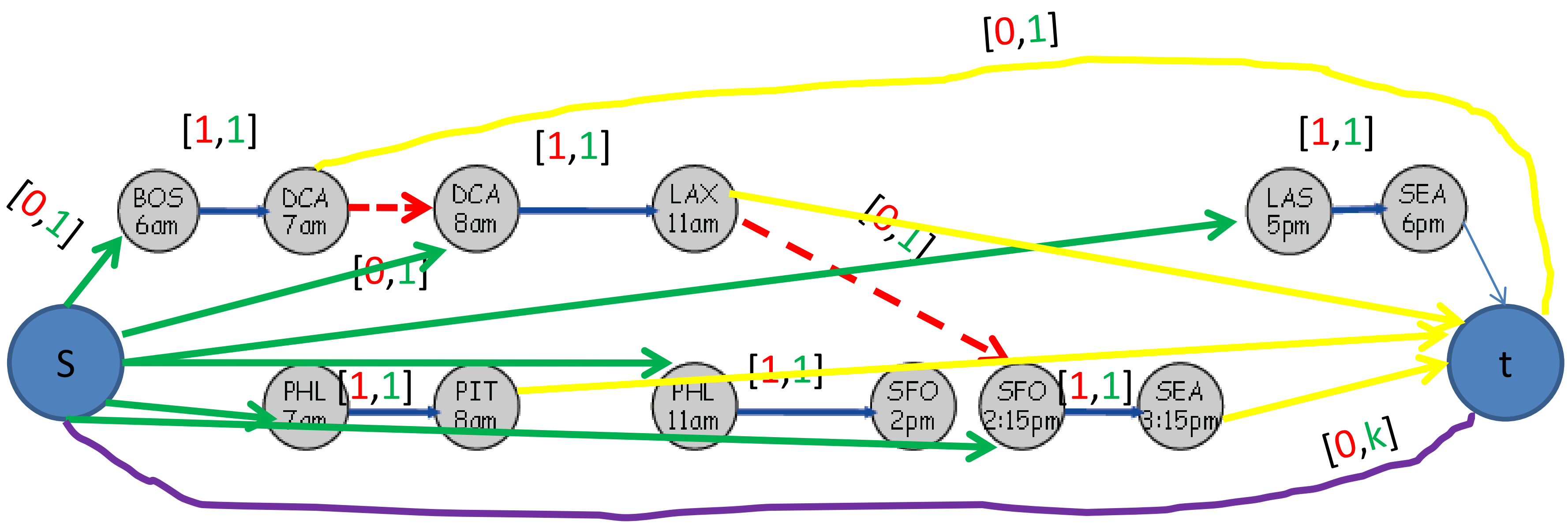
Designing the Algorithm

- The edge set of G is defined as follows.
  - For each j, there is an **edge (vj, t)** with a lower bound of 0 and a capacity of 1. (Any plane can end the day with flight j.)



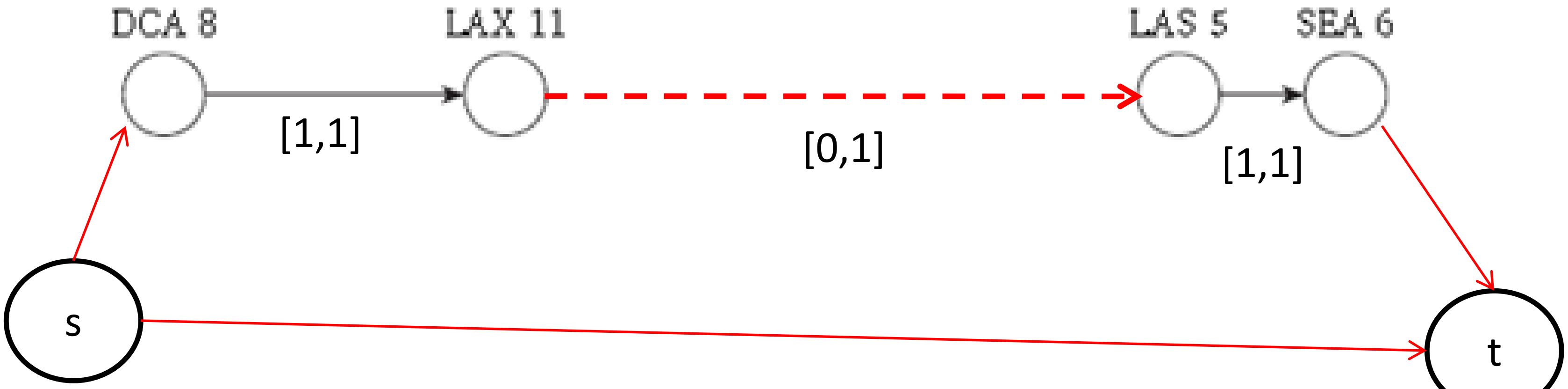
Designing the Algorithm

- The edge set of G is defined as follows.
  - There is an **edge (s, t)** with lower bound 0 and capacity k. (If we have extra planes, we don't need to use them for any of the flights.)



Analysis

There is a way to perform all flights using at most k planes if and only if there is a feasible circulation in the network G.

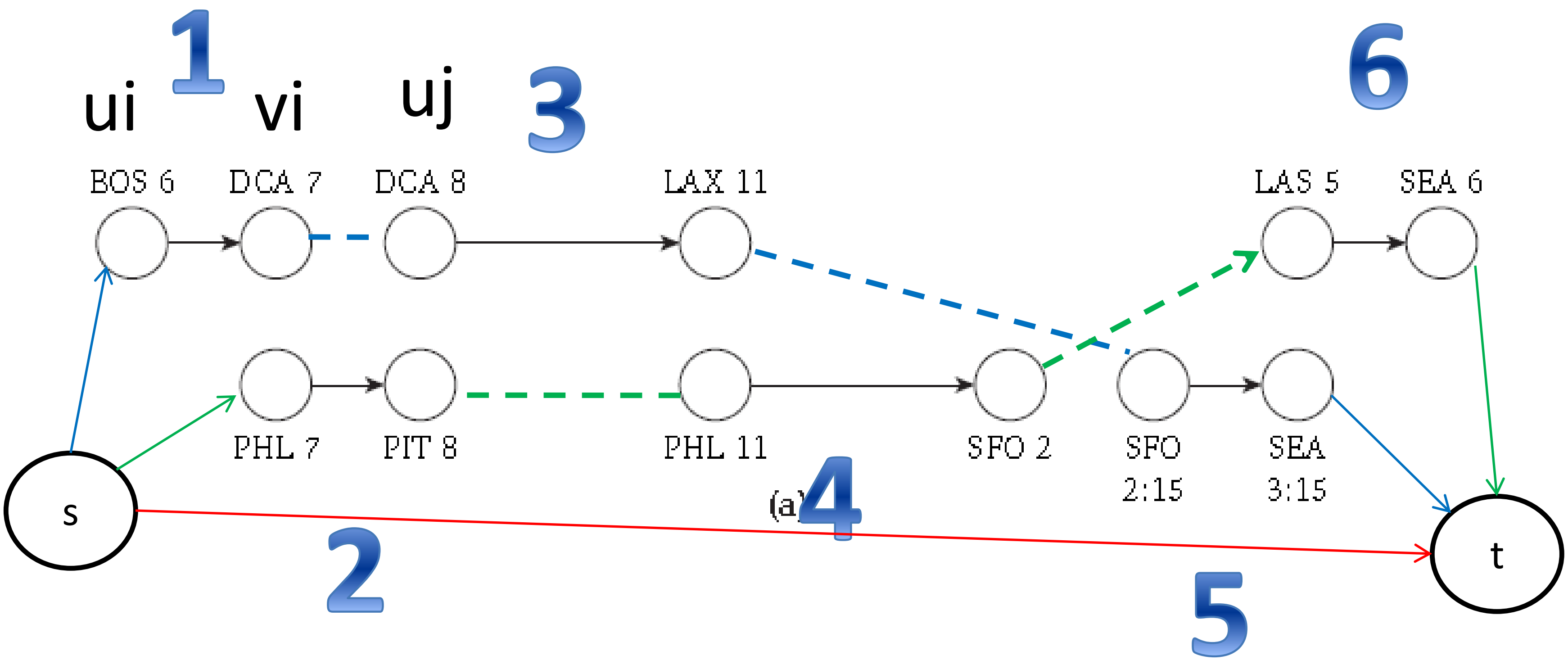


Feasible circulation means supply is equal to the demand

Proof

- Consider a feasible circulation in the network G.
- Suppose that k’ units of flow are sent on edges other than (s, t). Since all other edges have a capacity bound of 1, and the circulation is integer-valued, each such edge that carries flow has exactly one unit of flow on it.
- This flow can be converted to collection of paths.

Proof


1
ui
vi
uj
3
6
BOS 6
DCA 7
DCA 8
LAX 11
LAS 5
SEA 6
PHL 7
PIT 8
PHL 11
SFO 2
SFO
2:15
SEA
3:15
s
(a)
4
t
2
5

Proof

- Consider an edge (s, ui) that carries one unit of flow. It follows by conservation that (ui , vi) carries one unit of flow, and that there is a unique edge out of vi that carries one unit of flow.

- If we continue in this way, we construct a path P from s to t, so that each edge on this path carries one unit of flow.

Proof

- We can apply this construction to each edge of the form (s, uj) carrying one unit of flow; in this way, we produce k’ paths from s to t, each consisting of edges that carry one unit of flow.
-
- Now, for each path P we create in this way, we can assign a single plane to perform all the flights contained in this path.

THANK YOU