

Part I

Airline Scheduling

Airline Scheduling

Problem

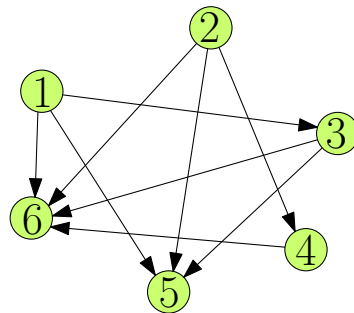
Given information about flights that an airline needs to provide, generate a profitable schedule.

1. Input: detailed information about “legs” of flight.
2. \mathcal{F} : set of flights by
3. Purpose: find minimum # airplanes needed.

Example

(i) a set \mathcal{F} of flights that have to be served, and (ii) the corresponding graph G representing these flights.

- 1: Boston (depart 6 A.M.) - Washington DC (arrive 7 A.M.),
- 2: Urbana (depart 7 A.M.) - Champaign (arrive 8 A.M.)
- 3: Washington (depart 8 A.M.) - Los Angeles (arrive 11 A.M.)
- 4: Urbana (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.).



(i)

(ii)

Flight scheduling...

1. Use same airplane for two segments i and j :
 - (a) destination of i is the origin of the segment j ,
 - (b) there is enough time in between the two flights.
2. Also, airplane can fly from $\text{dest}(i)$ to $\text{origin}(j)$ (assuming time constraints are satisfied).

Example

As a concrete example, consider the flights:

- Boston (depart 6 A.M.) - Washington D.C. (arrive 7 A.M.),
- Washington (depart 8 A.M.) - Los Angeles (arrive 11 A.M.)
- Las Vegas (depart 5 P.M.) - Seattle (arrive 6 P.M.)

This schedule can be served by a single airplane by adding the leg “Los Angeles (depart 12 noon)- Las Vegas (1 P.M.)” to this schedule.

Modeling the problem

1. model the feasibility constraints by a graph.
2. \mathbf{G} : directed graph over flight legs.
3. For i and j (legs), $(i \rightarrow j) \in \mathbf{E}(\mathbf{G}) \iff$ same airplane can serve both i and j .
4. \mathbf{G} is acyclic.
5. Q: Can required legs can be served using only k airplanes?

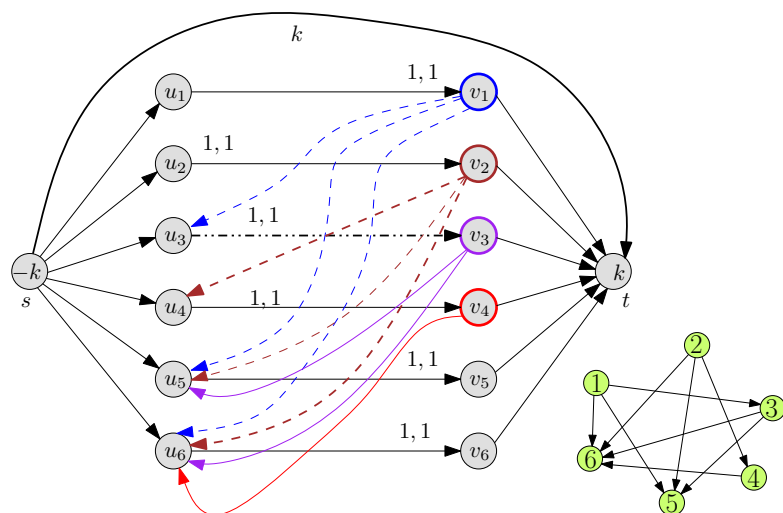
Solution

1. Reduction to computation of circulation.
2. Build graph \mathbf{H} .
3. \forall leg i , two new vertices $u_i, v_i \in \mathbf{VH}$.
 s : source vertex. t : sink vertex.
4. Set demand at t to k , Demand at s to be $-k$.
5. Each flight must be served. New edge $e_i = (u_i \rightarrow v_i)$, for leg i .
Also $\ell(e_i) = 1$ and $c(e_i) = 1$.
6. If same plane can so i and j (i.e., $(i \rightarrow j) \in \mathbf{E}(\mathbf{G})$) then add edge $(v_i \rightarrow u_j)$ with capacity 1 to \mathbf{H} .
7. Since any airplane can start the day with flight i : add an edge $(s \rightarrow u_i)$ with capacity 1 to \mathbf{H} , $\forall i$.
8. Add edge $(v_j \rightarrow t)$ with capacity 1 to \mathbf{G} , $\forall j$.
9. Overflow airplanes: "overflow" edge $(s \rightarrow t)$ with capacity k .

Let \mathbf{H} denote the resulting graph.

Example of resulting graph

The resulting graph \mathbf{H} for the instance of airline scheduling show before.



Lemma

Lemma

\exists way perform all flights of $\mathcal{F} \leq k$ planes $\iff \exists$ circulation in \mathbf{H} .

Proof.

1. Given feasible solution \rightarrow translate into valid circulation.
2. Given feasible circulation...
3. ... extract paths from flow.
4. ... every path is a plane.

□

Extensions and limitations

1. a lot of other considerations:
 - (i) airplanes have to undergo long term maintenance treatments every once in awhile,
 - (ii) one needs to allocate crew to these flights,
 - (iii) schedule differ between days, and
 - (iv) ultimately we interested in maximizing revenue.
2. Network flow is used in practice, real world problems are complicated, and network flow can capture only a few aspects.
3. ... a good starting point.

Part II

Image Segmentation

Image Segmentation

Input is an image.

Partition image into background and foreground.



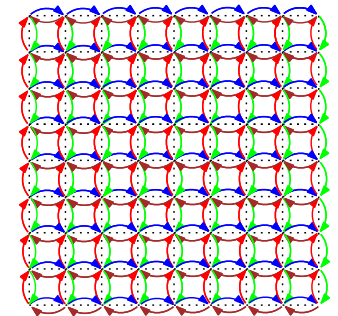
(i)

(ii)

The (i) input image, and (ii) a possible segmentation of the image.

What is the input...

1. Input is a bitmap on a grid.
2. Every grid node represents a pixel
3. Convert grid into a directed graph \mathbf{G} ,
4. Input:
 - (i) $\mathbf{N} \times \mathbf{N}$ bitmap.
 $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.
 - (ii) \forall pixel i : foreground value $f_i \geq 0$.
 - (iii) \forall pixel i : background value b_i .
 - (iv) $\forall i, j$ adjacent: separation penalty p_{ij} .
(we assume that $p_{ij} = p_{ji}$)



Problem statement

Problem

Given input as above, partition \mathbf{V} (the set of pixels) into two disjoint subsets \mathbf{F} and \mathbf{B} , such that

$$q(\mathbf{F}, \mathbf{B}) = \sum_{i \in \mathbf{F}} f_i + \sum_{i \in \mathbf{B}} b_i - \sum_{(i,j) \in \mathbf{E}, |F \cap \{i,j\}|=1} p_{ij}.$$

is maximized.

Rewrite $q(\mathbf{F}, \mathbf{B})$ as:

$$\begin{aligned} q(\mathbf{F}, \mathbf{B}) &= \sum_{i \in \mathbf{F}} f_i + \sum_{j \in \mathbf{B}} b_j - \sum_{(i,j) \in \mathbf{E}, |F \cap \{i,j\}|=1} p_{ij} \\ &= \sum_{i \in \mathbf{V}} (f_i + b_i) - \left(\sum_{i \in \mathbf{B}} f_i + \sum_{j \in \mathbf{F}} b_j + \sum_{(i,j) \in \mathbf{E}, |F \cap \{i,j\}|=1} p_{ij} \right). \end{aligned}$$

Restating problem...

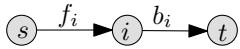
Maximizing:

$$q(\mathbf{F}, \mathbf{B}) = \sum_{i \in \mathbf{V}} (f_i + b_i) - \left(\sum_{i \in \mathbf{B}} f_i + \sum_{j \in \mathbf{F}} b_j + \sum_{(i,j) \in \mathbf{E}, |F \cap \{i,j\}|=1} p_{ij} \right).$$

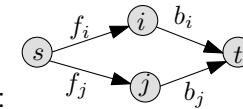
Equivalent to minimizing $u(\mathbf{F}, \mathbf{B})$:

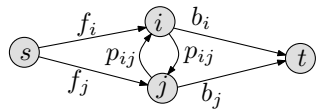
$$u(\mathbf{F}, \mathbf{B}) = \sum_{i \in \mathbf{B}} f_i + \sum_{j \in \mathbf{F}} b_j + \sum_{(i,j) \in \mathbf{E}, |F \cap \{i,j\}|=1} p_{ij}. \quad (1)$$

Solution continued...

1. Compute a minimum cut in a graph. Price = $u(\mathbf{F}, \mathbf{B})$.
2. A toy example: 
3. two possible cuts in the graph:
 - (i) $(\{s, i\}, \{t\})$: price b_i .
 - (ii) $(\{s\}, \{i, t\})$: price f_i . In particular,
4. Every path of length 2 from s to t forces mincut to choose one of edges.
Mincut "prefers" the edge with lower price.

Solution continued...



1. Two pixel bitmap:
2. Captures background/foreground prices. But... ignores separation penalties...
3. 
4. Price of cut in graph is corresponding value of $u(\mathbf{F}, \mathbf{B})$.
5. mincut-cut in the resulting graph would corresponds to the required segmentation.

Recap...

1. Given directed grid graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.
2. \mathbf{s}, \mathbf{y} : add two special source and sink vertices.
3. $\forall i \in \mathbf{V}_i$: add edge $\mathbf{e}_i = (\mathbf{s} \rightarrow i)$.
 $\mathbf{c}(\mathbf{e}_i) = \mathbf{f}_i$.
4. Add $\mathbf{e}'_i = (j \rightarrow \mathbf{t})$ with capacity $\mathbf{c}(\mathbf{e}'_i) = \mathbf{b}_i$.
5. $\forall i, j$ adjacent:
assign the capacity \mathbf{p}_{ij} to the edges $(i \rightarrow j)$ and $(j \rightarrow i)$

\mathbf{H} : resulting graph.

Solution continues...

By the above discussion:

Lemma

A minimum cut (\mathbf{F}, \mathbf{B}) in \mathbf{H} minimizes $\mathbf{u}(\mathbf{F}, \mathbf{B})$.

Using the minimum-cut max-flow theorem, we have:

Theorem

One can solve the segmentation problem, in polynomial time, by computing the max flow in the graph \mathbf{H} .

Part III

Projection selection

Project Selection

1. company which can carry out some projects.
2. \mathbf{P} : set of possible projects.
3. $\forall i \in \mathbf{P}$: a **revenue** \mathbf{p}_i .
4. $\mathbf{p}_i > \mathbf{0}$ is a profitable project and $\mathbf{p}_i < \mathbf{0}$ is a losing project.
5. There is dependency between projects.
6. $\mathbf{G} = (\mathbf{P}, \mathbf{E})$: $(i \rightarrow j) \in \mathbf{E}$ if and only if j is a prerequisite for i .

Definition

Definition

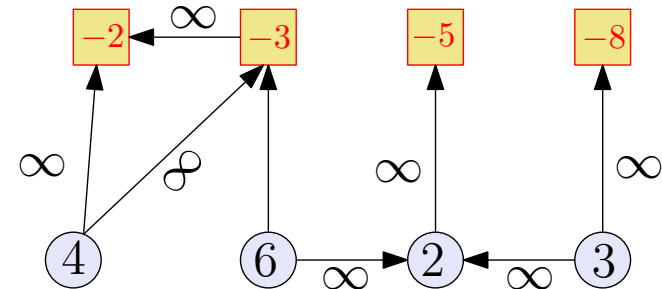
A set $X \subseteq P$ is **feasible** if for all $i \in X$, all the prerequisites of i are also in X . Formally, for all $i \in X$, with an edge $(i \rightarrow j) \in E$, we have $j \in X$.

The **profit** associated with a set of projects $X \subseteq P$ is $\text{profit}(X) = \sum_{i \in X} p_i$.

Problem - Project Selection Problem

Select a feasible set of projects maximizing the overall profit.

Project selection example

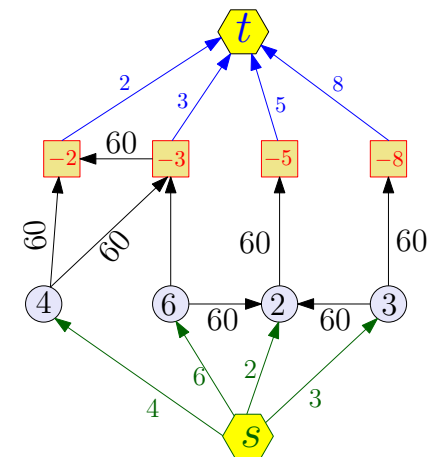


The reduction

1. Use mincut again.
2. Add s and t to G .
3. Perform the following modifications:
4. $\forall i \in P$ with $p_i > 0$: add edge $e_i = (s \rightarrow i)$ with $c(e_i) = p_i$.
5. $\forall j \in P$ with $p_j < 0$: add edge $e'_j = (j \rightarrow t)$. Set $c(e'_j) = -p_j$.
6. $C = \sum_{i \in P, p_i > 0} p_i$: upper bound on profit.
7. Set capacity of all original (dependency) edges in G to $4C$.

Let H denote the resulting network.

Example: Resulting network



Solution continued

1. $X \subseteq P$: Set of feasible projects.
2. $X' = X \cup \{s\}$ and $Y' = (P \setminus X) \cup \{t\}$.
3. Consider the s - t cut (X', Y') in H .
4. No $E(G)$ is in (X', Y') since X is a feasible set.

Lemma

Lemma

$$c(X', Y') = C - \sum_{i \in X} p_i = C - \text{profit}(X).$$

Proof

1. The edges of H are either:
 - (i) original edges of G ,
 - (ii) emanating from s , and
 - (iii) edges entering t .
2. X feasible \implies no edges of type (i) in cut.
3. Edges entering t contribute:

$$\beta = \sum_{i \in X \text{ and } p_i < 0} -p_i.$$

Proof continued

Proof.

Edges leaving s contribute:

$$\begin{aligned} \gamma &= \sum_{i \notin X \text{ and } p_i > 0} p_i = \sum_{i \in P, p_i > 0} p_i - \sum_{i \in X \text{ and } p_i > 0} p_i \\ &= C - \sum_{i \in X \text{ and } p_i > 0} p_i, \end{aligned}$$

by the definition of C . The capacity of the cut (X', Y') is

$$\begin{aligned} \beta + \gamma &= \sum_{i \in X \text{ and } p_i < 0} (-p_i) + \left(C - \sum_{i \in X \text{ and } p_i > 0} p_i \right) \\ &= C - \sum_{i \in X} p_i = C - \text{profit}(X), \end{aligned}$$

□

Lemma

Lemma

If (X', Y') is a cut with capacity at most C in G , then the set $X = X' \setminus \{s\}$ is a feasible set of projects.

Namely, cuts (X', Y') of capacity $\leq C$ in H corresponds one-to-one to feasible sets which are profitable.

Proof.

Since $c(X', Y') \leq C$ it must not cut any of the edges of G , since the price of such an edge is $4C$. As such, X must be a feasible set. □

Result

Theorem

If (X', Y') is a minimum cut in H then $X = X' \setminus \{s\}$ is an optimum solution to the project selection problem. In particular, using network flow the optimal solution can be computed in polynomial time.

Proof.

Indeed, we use network flow to compute the minimum cut in the resulting graph H . Note, that it is quite possible that the most profitable project is still a net loss. \square