More Network Flow Applications

Lecture 19

November 6, 2014

Moving into integral flow.

Given an integral network flow $\mathbf{G}$, with $\mathbf{n}$ vertices and $\mathbf{m}$ edges, consider a maximum flow $\mathbf{f}$ in $\mathbf{G}$, such that $\mathbf{C} = |\mathbf{f}|$. Assume $\mathbf{f}$ is not integral. Finding a maximum flow $\mathbf{g}$ that is integral and such that $|\mathbf{f}| = |\mathbf{g}|$ can be done in (faster is better):

- **(A)** $\mathbf{O(n + m)}$ time.
- **(B)** $\mathbf{O(nm)}$ time.
- **(C)** $\mathbf{O(mC)}$ time.
- **(D)** $\mathbf{O(m^2)}$ time.
- **(E)** $\mathbf{O(n \log n + m)}$ time.
- **(F)** Flow my tears the policeman said.

Part I

Baseball Pennant Race

Pennant Race

TUESDAY, SEPTEMBER 10, 1996

San Francisco Chronicle

The Gate
Sports Online
▶ http://www.sfgate.com

SPORTING G

49ers, Young Get Big Brea

Quarterback m

By Gary Swan
Chronicle Staff Writer

The bye week has come at a perfect time for the 49ers and quarterback Steve Young. If they had a game next Sunday, there's a good chance Young would not play.

But the pulled groin muscle on his up-

Giants Officially Leave the NL West Race

By Nancy Gay
Chronicle Staff Writer

With the smack of another National League West bat 500 miles away, the Giants' run at the division title ended last night, just as they were handing the visiting St. Louis Cardinals an even bigger lead in the NL Central.

CARDINALS 6
GIANTS 2

In San Diego, Greg Vaughn's three-run homer in the eighth pushed the Padres over the Pirates and officially shoved the rest of the Giants' season into the background. On the heels of their tedious 6-2 loss before an announced crowd of 10,307 at Candlestick Park, the Giants fell 19½ games off the lead.

As it is, the worst the Padres (80-65) can finish is 80-82. The Giants have fallen to 59-83 with 20

Financing in Place For Giants' New Stadium
SEE PAGE B1, MAIN NEWS

games left; they cannot win 80 games. Coming off a miserable 2-8 mark on a three-city road trip that saw their road record drop to 27-47, the Giants were hoping to get off on the right foot in their longest homestand of the year (15 games, 14 days).

"Where we are, you're going to be eliminated sooner or later," Baker said quietly. "But it doesn't alter the fact that we've still got to play ball. You've still got to play hard, the fans come out to watch you play. You've got to play for the fact of loving to play, no matter where you are in the standings.

"You've got to play the role of spoiler, to not make it easier on

GIANTS: Page D5 Col. 3

Pennant Race: Example

Example

Team	Won	Left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	89	2

Can Boston win the pennant?

No, because Boston can win at most 91 games.

Pennant Race: Example

Example

Team	Won	Left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	89	2

Can Boston win the pennant?
No, because Boston can win at most 91 games.

Another Example

Example

Team	Won	Left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	90	2

Can Boston win the pennant?

Not clear unless we know what the remaining games are!

Another Example

Example

Team	Won	Left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	90	2

Can Boston win the pennant?
Not clear unless we know what the remaining games are!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	—	1	1	0
Baltimore	91	3	1	—	1	1
Toronto	91	3	1	1	—	1
Boston	90	2	0	1	1	—

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	—	1	1	0
Baltimore	91	3	1	—	1	1
Toronto	91	3	1	1	—	1
Boston	90	2	0	1	1	—

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	—	1	1	0
Baltimore	91	3	1	—	1	1
Toronto	91	3	1	1	—	1
Boston	90	2	0	1	1	—

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	—	1	1	0
Baltimore	91	3	1	—	1	1
Toronto	91	3	1	1	—	1
Boston	90	2	0	1	1	—

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	—	1	1	0
Baltimore	91	3	1	—	1	1
Toronto	91	3	1	1	—	1
Boston	90	2	0	1	1	—

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Refining the Example

Example

Team	Won	Left	NY	Bal	Tor	Bos
New York	92	2	–	1	1	0
Baltimore	91	3	1	–	1	1
Toronto	91	3	1	1	–	1
Boston	90	2	0	1	1	–

Can Boston win the pennant? Suppose Boston does

1. Boston wins both its games to get 92 wins
2. New York must lose both games; now both Baltimore and Toronto have at least 92
3. Winner of Baltimore-Toronto game has 93 wins!

Can Boston win the penant?

Team	Won	Left	NY	Bal	Tor	Bos
New York	3	6	—	2	3	1
Baltimore	5	4	2	—	1	1
Toronto	4	6	3	1	—	2
Boston	2	4	1	1	2	—

(A) Yes.

(B) No.

Abstracting the Problem

Given

1. A set of teams $\mathbf{S}$
2. For each $\mathbf{x} \in \mathbf{S}$, the current number of wins $\mathbf{w_x}$
3. For any $\mathbf{x}, \mathbf{y} \in \mathbf{S}$, the number of remaining games $\mathbf{g_{xy}}$ between $\mathbf{x}$ and $\mathbf{y}$
4. A team $\mathbf{z}$

Can $\mathbf{z}$ win the pennant?

Towards a Reduction

$\bar{z}$ can win the pennant if

1. $\bar{z}$ wins at least $\mathbf{m}$ games
2. no other team wins more than $\mathbf{m}$ games

Towards a Reduction

$\bar{z}$ can win the pennant if

1. $\bar{z}$ wins at least $\mathbf{m}$ games
    1. to maximize $\bar{z}$'s chances we make $\bar{z}$ win all its remaining games and hence $\mathbf{m} = \mathbf{w}_{\bar{z}} + \sum_{x \in S} \mathbf{g}_{x\bar{z}}$
2. no other team wins more than $\mathbf{m}$ games

Towards a Reduction

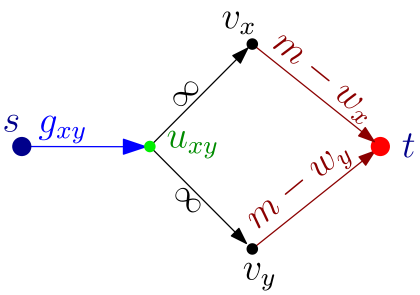
$\bar{z}$ can win the pennant if

1. $\bar{z}$ wins at least m games
   1. to maximize $\bar{z}$'s chances we make $\bar{z}$ win all its remaining games and hence $m = w_{\bar{z}} + \sum_{x \in S} g_{x\bar{z}}$
2. no other team wins more than m games
   1. for each $x, y \in S$ the g_{xy} games between them have to be *assigned* to either x or y.
   2. each team $x \neq \bar{z}$ can win at most $m - w_x - g_{x\bar{z}}$ remaining games

Is there an assignment of remaining games to teams such that no team $x \neq \bar{z}$ wins more than $m - w_x$ games?

Flow Network: The basic gadget

1. **s**: source
2. **t**: sink
3. **x**, **y**: two teams
4. $\mathbf{g_{xy}}$: number of games remaining between **x** and **y**.
5. $\mathbf{w_x}$: number of points **x** has.
6. **m**: maximum number of points **x** can win before team of interest is eliminated.

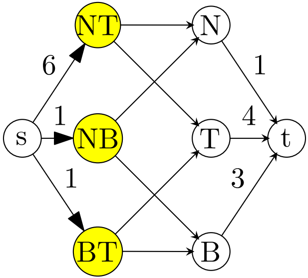


Flow Network: An Example

Can Boston win?

Team	Won	Left	NY	Bal	Tor	Bos
New York	90	11	–	1	6	4
Baltimore	88	6	1	–	1	4
Toronto	87	11	6	1	–	4
Boston	79	12	4	4	4	–

1. $\mathbf{m = 79 + 12 = 91}$: Boston can get at most **91** points.



Constructing Flow Network

Notations

1. $\mathbf{S}$: set of teams,
2. $\mathbf{w_x}$ wins for each team, and
3. $\mathbf{g_{xy}}$ games left between $\mathbf{x}$ and $\mathbf{y}$.
4. $\mathbf{m}$ be the maximum number of wins for $\bar{\mathbf{z}}$,
5. and $\mathbf{S'} = \mathbf{S} \setminus \{\bar{\mathbf{z}}\}$.

Reduction

Construct the flow network $\mathbf{G}$ as follows

1. One vertex $\mathbf{v_x}$ for each team $\mathbf{x} \in \mathbf{S'}$, one vertex $\mathbf{u_{xy}}$ for each pair of teams $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{S'}$
2. A new source vertex $\mathbf{s}$ and sink $\mathbf{t}$
3. Edges $(\mathbf{u_{xy}}, \mathbf{v_x})$ and $(\mathbf{u_{xy}}, \mathbf{v_y})$ of capacity ∞
4. Edges $(\mathbf{s}, \mathbf{u_{xy}})$ of capacity $\mathbf{g_{xy}}$
5. Edges $(\mathbf{v_x}, \mathbf{t})$ of capacity equal $\mathbf{m} - \mathbf{w_x}$

Correctness of reduction

Theorem

$\mathbf{G'}$ *has a maximum flow of value* $\mathbf{g^*} = \sum_{\mathbf{x,y} \in \mathbf{S'}} \mathbf{g_{xy}}$ *if and only if* $\overline{\mathbf{z}}$ *can win the most number of games (including possibly tie with other teams).*

Proof of Correctness

Proof.

Existence of $\mathbf{g}^*$ flow $\Rightarrow$ $\bar{z}$ wins pennant

1. An integral flow saturating edges out of $\mathbf{s}$, ensures that each remaining game between $\mathbf{x}$ and $\mathbf{y}$ is added to win total of either $\mathbf{x}$ or $\mathbf{y}$
2. Capacity on $(\mathbf{v_x}, \mathbf{t})$ edges ensures that no team wins more than $\mathbf{m}$ games

Conversely, $\bar{z}$ wins pennant $\Rightarrow$ flow of value $\mathbf{g}^*$

1. Scenario determines flow on edges; if $\mathbf{x}$ wins $\mathbf{k}$ of the games against $\mathbf{y}$, then flow on $(\mathbf{u_{xy}}, \mathbf{v_x})$ edge is $\mathbf{k}$ and on $(\mathbf{u_{xy}}, \mathbf{v_y})$ edge is $\mathbf{g_{xy}} - \mathbf{k}$ $\square$

Proof that $\bar{z}$ cannot with the pennant

1. Suppose $\bar{z}$ cannot win the pennant since $\mathbf{g}^* < \mathbf{g}$. How do we *prove* to some one *compactly* that $\bar{z}$ cannot win the pennant?
2. Show them the min-cut in the reduction flow network!
3. See text book for a natural interpretation of the min-cut as a certificate.

Proof that $\bar{\mathbf{z}}$ cannot with the pennant

1. Suppose $\bar{\mathbf{z}}$ cannot win the pennant since $\mathbf{g}^* < \mathbf{g}$. How do we *prove* to some one *compactly* that $\bar{\mathbf{z}}$ cannot win the pennant?
2. Show them the min-cut in the reduction flow network!
3. See text book for a natural interpretation of the min-cut as a certificate.

Proof that $\bar{z}$ cannot with the pennant

1. Suppose $\bar{z}$ cannot win the pennant since $\mathbf{g}^* < \mathbf{g}$. How do we *prove* to some one *compactly* that $\bar{z}$ cannot win the pennant?
2. Show them the min-cut in the reduction flow network!
3. See text book for a natural interpretation of the min-cut as a certificate.

The biggest loser?

Given an input as above for the pennant competition, deciding if a team can come in the last place can be done in

(A) Can be done using the same reduction as just seen.

(B) Can not be done using the same reduction as just seen.

(C) Can be done using flows but we need lower bounds on the flow, instead of upper bounds.

(D) The problem is **NP-Hard** and requires exponential time.

(E) Can be solved by negating all the numbers, and using the above reduction.

(F) Can be solved efficiently only by running a reality show on the problem.

Part II

An Application of Min-Cut to Project Scheduling

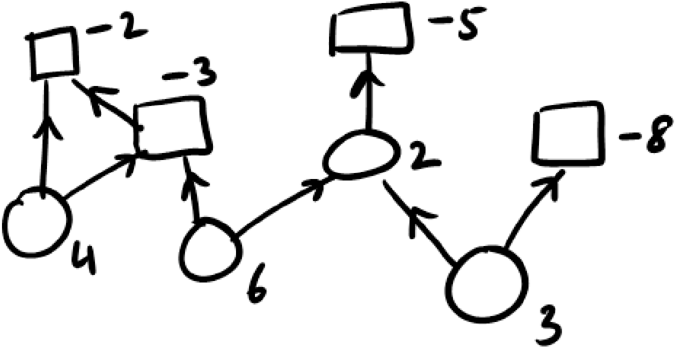
Project Scheduling

Problem:

1. $\mathbf{n}$ projects/tasks $1, 2, \ldots, \mathbf{n}$
2. *dependencies* between projects: $\mathbf{i}$ depends on $\mathbf{j}$ implies $\mathbf{i}$ cannot be done unless $\mathbf{j}$ is done. dependency graph is *acyclic*
3. each project $\mathbf{i}$ has a cost/profit $\mathbf{p_i}$
    1. $\mathbf{p_i} < 0$ implies $\mathbf{i}$ requires a cost of $-\mathbf{p_i}$ units
    2. $\mathbf{p_i} > 0$ implies that $\mathbf{i}$ generates $\mathbf{p_i}$ profit

Goal: Find projects to do so as to *maximize* profit.

Example



Notation

For a set $\mathbf{A}$ of projects:

1. $\mathbf{A}$ is a *valid* solution if $\mathbf{A}$ is *dependency closed*, that is for every $\mathbf{i} \in \mathbf{A}$, all projects that $\mathbf{i}$ depends on are also in $\mathbf{A}$.
2. $\mathbf{profit(A)} = \sum_{\mathbf{i} \in \mathbf{A}} \mathbf{p_i}$. Can be negative or positive.

Goal: find valid $\mathbf{A}$ to maximize $\mathbf{profit(A)}$.

Notation

For a set $\mathbf{A}$ of projects:

1. $\mathbf{A}$ is a *valid* solution if $\mathbf{A}$ is *dependency closed*, that is for every $\mathbf{i} \in \mathbf{A}$, all projects that $\mathbf{i}$ depends on are also in $\mathbf{A}$.
2. $\mathbf{profit(A)} = \sum_{\mathbf{i} \in \mathbf{A}} \mathbf{p_i}$. Can be negative or positive.

Goal: find valid $\mathbf{A}$ to maximize $\mathbf{profit(A)}$.

Notation

For a set $\mathbf{A}$ of projects:

1. $\mathbf{A}$ is a *valid* solution if $\mathbf{A}$ is *dependency closed*, that is for every $\mathbf{i} \in \mathbf{A}$, all projects that $\mathbf{i}$ depends on are also in $\mathbf{A}$.
2. $\mathbf{profit(A)} = \sum_{\mathbf{i} \in \mathbf{A}} \mathbf{p_i}$. Can be negative or positive.

Goal: find valid $\mathbf{A}$ to maximize $\mathbf{profit(A)}$.

Idea: Reduction to Minimum-Cut

Finding a set of projects is partitioning the projects into two sets: those that are done and those that are not done.

Can we express this is a minimum cut problem?

Several issues:

1. We are interested in maximizing profit but we can solve minimum cuts.
2. We need to convert negative profits into positive capacities.
3. Need to ensure that chosen projects is a valid set.
4. The cut value captures the profit of the chosen set of projects.

Idea: Reduction to Minimum-Cut

Finding a set of projects is partitioning the projects into two sets: those that are done and those that are not done.

Can we express this is a minimum cut problem?

Several issues:

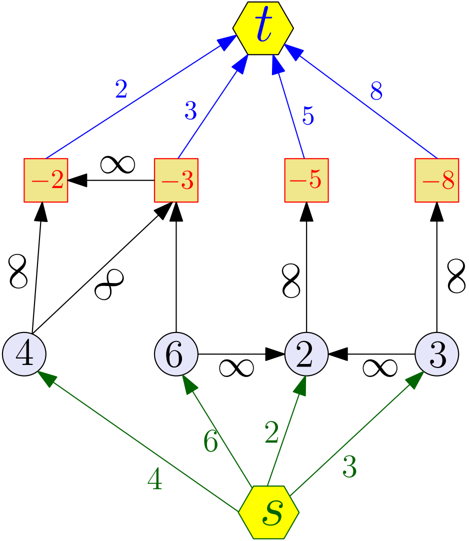
1. We are interested in maximizing profit but we can solve minimum cuts.
2. We need to convert negative profits into positive capacities.
3. Need to ensure that chosen projects is a valid set.
4. The cut value captures the profit of the chosen set of projects.

Reduction to Minimum-Cut

Note: We are reducing a *maximization* problem to a *minimization* problem.

1. projects represented as nodes in a graph
2. if $\mathbf{i}$ depends on $\mathbf{j}$ then $(\mathbf{i}, \mathbf{j})$ is an edge
3. add source $\mathbf{s}$ and sink $\mathbf{t}$
4. for each $\mathbf{i}$ with $\mathbf{p_i} > \mathbf{0}$ add edge $(\mathbf{s}, \mathbf{i})$ with capacity $\mathbf{p_i}$
5. for each $\mathbf{i}$ with $\mathbf{p_i} < \mathbf{0}$ add edge $(\mathbf{i}, \mathbf{t})$ with capacity $-\mathbf{p_i}$
6. for each dependency edge $(\mathbf{i}, \mathbf{j})$ put capacity ∞ (more on this later)

Reduction: Flow Network Example



Reduction contd

Algorithm:

1. form graph as in previous slide
2. compute $\mathbf{s}$-$\mathbf{t}$ minimum cut $\mathbf{(A, B)}$
3. output the projects in $\mathbf{A} - \{\mathbf{s}\}$

Understanding the Reduction

Let $\mathbf{C} = \sum_{i:p_i>0} p_i$: maximum possible profit.

Observation: The minimum s-t cut value is $\leq C$. Why?

Lemma

Suppose (A, B) is an s-t cut of finite capacity (no ∞) edges. Then projects in $A - \{s\}$ are a valid *solution.*

Proof.

If $A - \{s\}$ is not a valid solution then there is a project $i \in A$ and a project $j \notin A$ such that i depends on j

Since (i, j) capacity is ∞, implies (A, B) capacity is ∞, contradicting assumption. $\square$

Understanding the Reduction

Let $\mathbf{C} = \sum_{i:p_i>0} \mathbf{p_i}$: maximum possible profit.

Observation: The minimum $\mathbf{s}$-$\mathbf{t}$ cut value is $\leq \mathbf{C}$. Why?

Lemma

Suppose $(\mathbf{A}, \mathbf{B})$ is an $\mathbf{s}$-$\mathbf{t}$ cut of finite capacity (no ∞) edges. Then projects in $\mathbf{A} - \{\mathbf{s}\}$ are a valid *solution.*

Proof.

If $\mathbf{A} - \{\mathbf{s}\}$ is not a valid solution then there is a project $\mathbf{i} \in \mathbf{A}$ and a project $\mathbf{j} \notin \mathbf{A}$ such that $\mathbf{i}$ depends on $\mathbf{j}$

Since $(\mathbf{i}, \mathbf{j})$ capacity is ∞, implies $(\mathbf{A}, \mathbf{B})$ capacity is ∞, contradicting assumption. $\square$

Understanding the Reduction

Let $\mathbf{C} = \sum_{i:p_i > 0} \mathbf{p_i}$: maximum possible profit.

Observation: The minimum $\mathbf{s}$-$\mathbf{t}$ cut value is $\leq \mathbf{C}$. Why?

Lemma

Suppose $(\mathbf{A}, \mathbf{B})$ *is an* $\mathbf{s}$-$\mathbf{t}$ *cut of finite capacity (no* ∞*) edges. Then projects in* $\mathbf{A} - \{\mathbf{s}\}$ *are a* valid *solution.*

Proof.

If $\mathbf{A} - \{\mathbf{s}\}$ is not a valid solution then there is a project $\mathbf{i} \in \mathbf{A}$ and a project $\mathbf{j} \notin \mathbf{A}$ such that $\mathbf{i}$ depends on $\mathbf{j}$

Since $(\mathbf{i}, \mathbf{j})$ capacity is ∞, implies $(\mathbf{A}, \mathbf{B})$ capacity is ∞, contradicting assumption. $\square$

Understanding the Reduction

Let $\mathbf{C} = \sum_{\mathbf{i}:\mathbf{p_i}>0} \mathbf{p_i}$: maximum possible profit.

Observation: The minimum $\mathbf{s}$-$\mathbf{t}$ cut value is $\leq \mathbf{C}$. Why?

Lemma

Suppose $(\mathbf{A}, \mathbf{B})$ is an $\mathbf{s}$-$\mathbf{t}$ cut of finite capacity (no ∞) edges. Then projects in $\mathbf{A} - \{\mathbf{s}\}$ are a valid *solution.*

Proof.

If $\mathbf{A} - \{\mathbf{s}\}$ is not a valid solution then there is a project $\mathbf{i} \in \mathbf{A}$ and a project $\mathbf{j} \notin \mathbf{A}$ such that $\mathbf{i}$ depends on $\mathbf{j}$

Since $(\mathbf{i}, \mathbf{j})$ capacity is ∞, implies $(\mathbf{A}, \mathbf{B})$ capacity is ∞, contradicting assumption. $\square$

Understanding the Reduction

Let $\mathbf{C} = \sum_{i:p_i>0} \mathbf{p_i}$: maximum possible profit.

Observation: The minimum $\mathbf{s}$-$\mathbf{t}$ cut value is $\leq \mathbf{C}$. Why?

Lemma

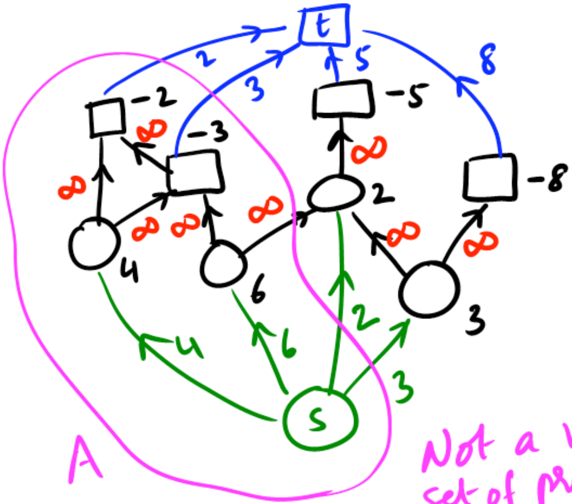
Suppose $(\mathbf{A}, \mathbf{B})$ *is an* $\mathbf{s}$-$\mathbf{t}$ *cut of finite capacity (no* ∞*) edges. Then projects in* $\mathbf{A} - \{\mathbf{s}\}$ *are a* valid *solution.*

Proof.

If $\mathbf{A} - \{\mathbf{s}\}$ is not a valid solution then there is a project $\mathbf{i} \in \mathbf{A}$ and a project $\mathbf{j} \notin \mathbf{A}$ such that $\mathbf{i}$ depends on $\mathbf{j}$

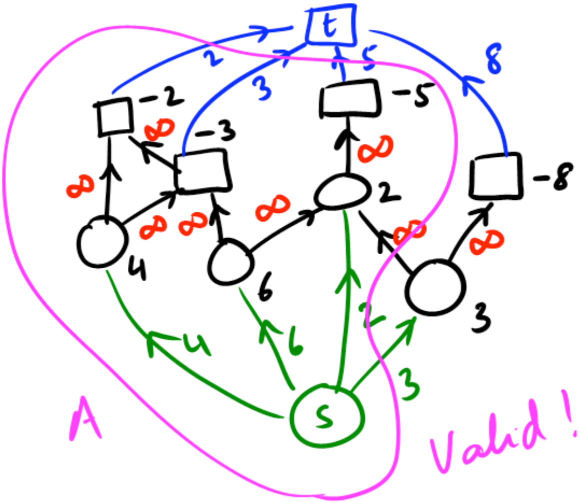
Since $(\mathbf{i}, \mathbf{j})$ capacity is ∞, implies $(\mathbf{A}, \mathbf{B})$ capacity is ∞, contradicting assumption. $\square$

Example



Not a valid set of projects

Example



Correctness of Reduction

Recall that for a set of projects $\mathbf{X}$, $\mathbf{profit(X)} = \sum_{i \in X} p_i$.

Lemma

Suppose (A, B) *is an* s-t *cut of finite capacity (no* ∞*) edges. Then* $c(A, B) = C - \text{profit}(A - \{s\})$.

Proof.

Edges in (A, B):

1. (s, i) for $i \in B$ and $p_i > 0$: capacity is p_i
2. (i, t) for $i \in A$ and $p_i < 0$: capacity is $-p_i$
3. cannot have ∞ edges

Correctness of Reduction

Recall that for a set of projects $\mathbf{X}$, $\mathbf{profit(X)} = \sum_{i \in X} \mathbf{p_i}$.

Lemma

Suppose $(\mathbf{A}, \mathbf{B})$ *is an* $\mathbf{s}$-$\mathbf{t}$ *cut of finite capacity (no* ∞*) edges. Then* $\mathbf{c(A, B) = C - profit(A - \{s\})}$.

Proof.

Edges in $(\mathbf{A}, \mathbf{B})$:

1. $(\mathbf{s}, \mathbf{i})$ for $\mathbf{i} \in \mathbf{B}$ and $\mathbf{p_i} > \mathbf{0}$: capacity is $\mathbf{p_i}$
2. $(\mathbf{i}, \mathbf{t})$ for $\mathbf{i} \in \mathbf{A}$ and $\mathbf{p_i} < \mathbf{0}$: capacity is $-\mathbf{p_i}$
3. cannot have ∞ edges

□

Correctness of Reduction

Recall that for a set of projects $\mathbf{X}$, $\mathbf{profit(X)} = \sum_{i \in X} \mathbf{p_i}$.

Lemma

Suppose $\mathbf{(A, B)}$ *is an* $\mathbf{s}$-$\mathbf{t}$ *cut of finite capacity (no* ∞*) edges. Then* $\mathbf{c(A, B) = C - profit(A - \{s\})}$.

Proof.

Edges in $\mathbf{(A, B)}$:

1. $\mathbf{(s, i)}$ for $\mathbf{i} \in \mathbf{B}$ and $\mathbf{p_i} > \mathbf{0}$: capacity is $\mathbf{p_i}$
2. $\mathbf{(i, t)}$ for $\mathbf{i} \in \mathbf{A}$ and $\mathbf{p_i} < \mathbf{0}$: capacity is $-\mathbf{p_i}$
3. cannot have ∞ edges

□

Proof contd

For project set $\mathbf{A}$ let

1. $\mathbf{cost(A) = \sum_{i \in A : p_i < 0} -p_i}$
2. $\mathbf{benefit(A) = \sum_{i \in A : p_i > 0} p_i}$
3. $\mathbf{profit(A) = benefit(A) - cost(A)}$.

Proof.

Let $\mathbf{A' = A \cup \{s\}}$.

$$
\begin{aligned}
\mathbf{c(A', B)} &= \mathbf{cost(A) + benefit(B)} \\
&= \mathbf{cost(A) - benefit(A) + benefit(A) + benefit(B)} \\
&= \mathbf{-profit(A) + C} \\
&= \mathbf{C - profit(A)}
\end{aligned}
$$

□

Correctness of Reduction contd

We have shown that if (A, B) is an s-t cut in G with finite capacity then

1. $A - \{s\}$ is a valid set of projects
2. $c(A, B) = C - \text{profit}(A - \{s\})$

Therefore a *minimum* s-t cut (A^*, B^*) gives a *maximum* profit set of projects $A^* - \{s\}$ since C is fixed.

Question: How can we use ∞ in a real algorithm?

Set capacity of ∞ arcs to $C + 1$ instead. Why does this work?

Correctness of Reduction contd

We have shown that if (A, B) is an s-t cut in G with finite capacity then

1. $A - \{s\}$ is a valid set of projects
2. $c(A, B) = C - \text{profit}(A - \{s\})$

Therefore a *minimum* s-t cut (A^*, B^*) gives a *maximum* profit set of projects $A^* - \{s\}$ since C is fixed.

Question: How can we use ∞ in a real algorithm?

Set capacity of ∞ arcs to $C + 1$ instead. Why does this work?

Correctness of Reduction contd

We have shown that if (A, B) is an s-t cut in G with finite capacity then

1. $A - \{s\}$ is a valid set of projects
2. $c(A, B) = C - \text{profit}(A - \{s\})$

Therefore a *minimum* s-t cut (A^*, B^*) gives a *maximum* profit set of projects $A^* - \{s\}$ since C is fixed.

Question: How can we use ∞ in a real algorithm?

Set capacity of ∞ arcs to $C + 1$ instead. Why does this work?

Correctness of Reduction contd

We have shown that if (A, B) is an s-t cut in G with finite capacity then

1. $A - \{s\}$ is a valid set of projects
2. $c(A, B) = C - \text{profit}(A - \{s\})$

Therefore a *minimum* s-t cut (A^*, B^*) gives a *maximum* profit set of projects $A^* - \{s\}$ since C is fixed.

Question: How can we use ∞ in a real algorithm?

Set capacity of ∞ arcs to $C + 1$ instead. Why does this work?

Shortest path always present?

Let $\mathbf{G}$ be an directed graph, and let $\mathbf{\Pi} = \left\{\pi_1, \ldots, \pi_|\mathbf{k}\right\}$ be the (largest) set of edge disjoint paths in $\mathbf{G}$ from $\mathbf{s}$ to $\mathbf{t}$, computed using network flow.

- **(A)** The shortest path in $\mathbf{G}$ must be one of the paths in $\mathbf{\Pi}$.
- **(B)** The shortest path in $\mathbf{G}$ must intersects exactly <u>one</u> of the paths in $\mathbf{\Pi}$.
- **(C)** The shortest path in $\mathbf{G}$ must intersects <u>all</u> of the paths in $\mathbf{\Pi}$.
- **(D)** The shortest path in $\mathbf{G}$ must intersects at least one of the paths in $\mathbf{\Pi}$, and it can intersect all of them.

Part III

Extensions to Maximum-Flow Problem

Lower Bounds and Costs

Two generalizations:

1. flow satisfies $\mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$. suppose we are given *lower bounds* $\boldsymbol{\ell}\mathbf{(e)}$ for each $\mathbf{e}$. can we find a flow such that $\boldsymbol{\ell}\mathbf{(e)} \leq \mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$?
2. suppose we are given a cost $\mathbf{w(e)}$ for each edge. cost of routing flow $\mathbf{f(e)}$ on edge $\mathbf{e}$ is $\mathbf{w(e)f(e)}$. can we (efficiently) find a flow (of at least some given quantity) at minimum cost?

Many applications.

Lower Bounds and Costs

Two generalizations:

1. flow satisfies $\mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$. suppose we are given *lower bounds* $\ell(\mathbf{e})$ for each $\mathbf{e}$. can we find a flow such that $\ell(\mathbf{e}) \leq \mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$?
2. suppose we are given a cost $\mathbf{w(e)}$ for each edge. cost of routing flow $\mathbf{f(e)}$ on edge $\mathbf{e}$ is $\mathbf{w(e)f(e)}$. can we (efficiently) find a flow (of at least some given quantity) at minimum cost?

Many applications.

Lower Bounds and Costs

Two generalizations:

1. flow satisfies $\mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$. suppose we are given *lower bounds* $\ell(\mathbf{e})$ for each $\mathbf{e}$. can we find a flow such that $\ell(\mathbf{e}) \leq \mathbf{f(e)} \leq \mathbf{c(e)}$ for all $\mathbf{e}$?
2. suppose we are given a cost $\mathbf{w(e)}$ for each edge. cost of routing flow $\mathbf{f(e)}$ on edge $\mathbf{e}$ is $\mathbf{w(e)f(e)}$. can we (efficiently) find a flow (of at least some given quantity) at minimum cost?

Many applications.

Flows with Lower Bounds

Definition

A flow in a network $\mathbf{G = (V, E)}$, is a function $\mathbf{f : E \to \mathbb{R}^{\geq 0}}$ such that

1. Capacity Constraint: For each edge $\mathbf{e}$, $\mathbf{f(e) \leq c(e)}$
2. Lower Bound Constraint: For each edge $\mathbf{e}$, $\mathbf{f(e) \geq \ell(e)}$
3. Conservation Constraint: For each vertex $\mathbf{v}$

$$\sum_{\mathbf{e} \text{ into } \mathbf{v}} \mathbf{f(e)} = \sum_{\mathbf{e} \text{ out of } \mathbf{v}} \mathbf{f(e)}$$

Question: Given $\mathbf{G}$ and $\mathbf{c(e)}$ and $\mathbf{\ell(e)}$ for each $\mathbf{e}$, is there a flow?
As difficult as finding an $\mathbf{s}$-$\mathbf{t}$ maximum-flow without lower bounds!

Flows with Lower Bounds

Definition

A flow in a network $\mathbf{G = (V, E)}$, is a function $\mathbf{f : E \rightarrow} \mathbb{R}^{\geq 0}$ such that

1. Capacity Constraint: For each edge $\mathbf{e}$, $\mathbf{f(e) \leq c(e)}$
2. Lower Bound Constraint: For each edge $\mathbf{e}$, $\mathbf{f(e) \geq \ell(e)}$
3. Conservation Constraint: For each vertex $\mathbf{v}$

$$\sum_{\mathbf{e \text{ into } v}} \mathbf{f(e)} = \sum_{\mathbf{e \text{ out of } v}} \mathbf{f(e)}$$

Question: Given $\mathbf{G}$ and $\mathbf{c(e)}$ and $\ell\mathbf{(e)}$ for each $\mathbf{e}$, is there a flow?

As difficult as finding an $\mathbf{s}$-$\mathbf{t}$ maximum-flow without lower bounds!

Flows with Lower Bounds

Definition

A flow in a network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, is a function $\mathbf{f} : \mathbf{E} \to \mathbb{R}^{\geq 0}$ such that

1. Capacity Constraint: For each edge $\mathbf{e}$, $\mathbf{f}(\mathbf{e}) \leq \mathbf{c}(\mathbf{e})$
2. Lower Bound Constraint: For each edge $\mathbf{e}$, $\mathbf{f}(\mathbf{e}) \geq \ell(\mathbf{e})$
3. Conservation Constraint: For each vertex $\mathbf{v}$

$$\sum_{\mathbf{e} \text{ into } \mathbf{v}} \mathbf{f}(\mathbf{e}) = \sum_{\mathbf{e} \text{ out of } \mathbf{v}} \mathbf{f}(\mathbf{e})$$

Question: Given $\mathbf{G}$ and $\mathbf{c}(\mathbf{e})$ and $\ell(\mathbf{e})$ for each $\mathbf{e}$, is there a flow? As difficult as finding an $\mathbf{s}$-$\mathbf{t}$ maximum-flow without lower bounds!

Flows with Lower Bounds

1. Flows with lower bounds can be reduced to standard maximum flow problem. See text book. Reduction goes via circulations.
2. If all bounds are integers then there is a flow that is integral. Useful in applications.

Combining max flows?

Given distinct max flows $\mathbf{f}$ and $\mathbf{g}$ in $\mathbf{G}$, the function $\mathbf{h(e) = (f(e) + g(e))/2}$, for all $\mathbf{e} \in \mathbf{E(G)}$, describes a valid max flow in $\mathbf{G}$. This is

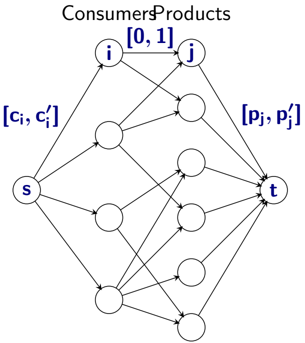
- **(A)** True.
- **(B)** False.

Survey Design

Application of Flows with Lower Bounds

1. Design survey to find information about $\mathbf{n_1}$ products from $\mathbf{n_2}$ customers.
2. Can ask customer questions only about products purchased in the past.
3. Customer can only be asked about at most $\mathbf{c_i'}$ products and at least $\mathbf{c_i}$ products.
4. For each product need to ask at east $\mathbf{p_i}$ consumers and at most $\mathbf{p_i'}$ consumers.

Reduction to Circulation



1. include edge (i, j) is customer i has bought product j
2. Add edge (t, s) with lower bound 0 and upper bound ∞.
    1. Consumer i is asked about product j if the integral flow on edge (i, j) is 1

Minimum Cost Flows

1. **Input:** Given a flow network $\mathbf{G}$ and also edge costs, $\mathbf{w(e)}$ for edge $\mathbf{e}$, and a flow requirement $\mathbf{F}$.
2. **Goal;** Find a *minimum cost* flow of value $\mathbf{F}$ from $\mathbf{s}$ to $\mathbf{t}$

Given flow $\mathbf{f : E \to R^+}$, cost of flow $= \sum_{\mathbf{e \in E}} \mathbf{w(e)f(e)}$.

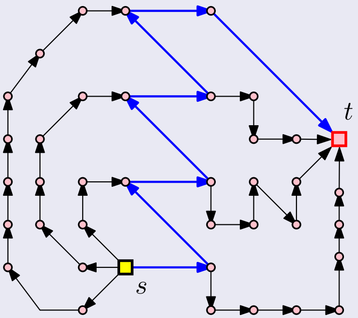
Minimum Cost Flow: Facts

1. problem can be solved efficiently in polynomial time
   1. $O(nm \log C \log(nW))$ time algorithm where C is maximum edge capacity and W is maximum edge cost
   2. $O(m \log n(m + n \log n))$ time strongly polynomial time algorithm
2. for integer capacities there is always an optimum solutions in which flow is integral

How much damage can a single path cause?

Consider the following network. All the edges have capacity **1**. Clearly the maximum flow in this network has value **4**.

The network



Why removing the shortest path might ruin everything

1. However... The shortest path between **s** and **t** is the blue path.
2. And if we remove the shortest path, **s** and **t** become disconnected, and the maximum flow drop to **0**.

Notes

Notes

Notes

Notes