ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Special Topics on Algorithms

Applications of Linear and Integer Programming

Vangelis Markakis   Ioannis Milis
George Zois

Linear Programming

Quick applications of LP:

1. Flows in networks

2. Matching in bipartite graphs

Flows in Networks

Recall the max flow problem:

Consider a graph G = (V, E), with a source node s $\in$ V, and a sink node t $\in$ V

Capacity constraints: for every edge e $\in$ E, there is a capacity c_e

A feasible flow is an assignment of a flow f_e to every edge so that

1. $f_e \leq c_e$
2. For every node other than source and sink:
   incoming flow = outgoing flow (preservation of flow)

Goal: find a feasible flow so as to maximize the total amount of flow coming out of s (or equivalently going into t)

Flow going out of s: $\sum_{(s,u)\in E} f_{su}$

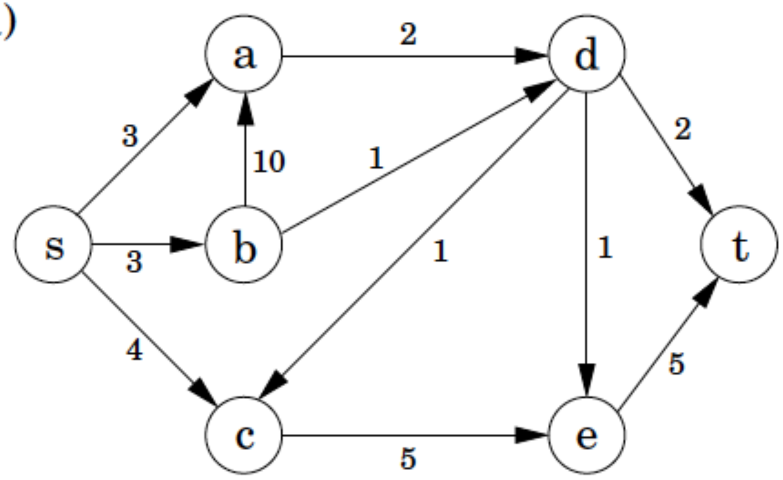
By preservation of flow this equals: $\sum_{(u,t)\in E} f_{ut}$

Flows in Networks

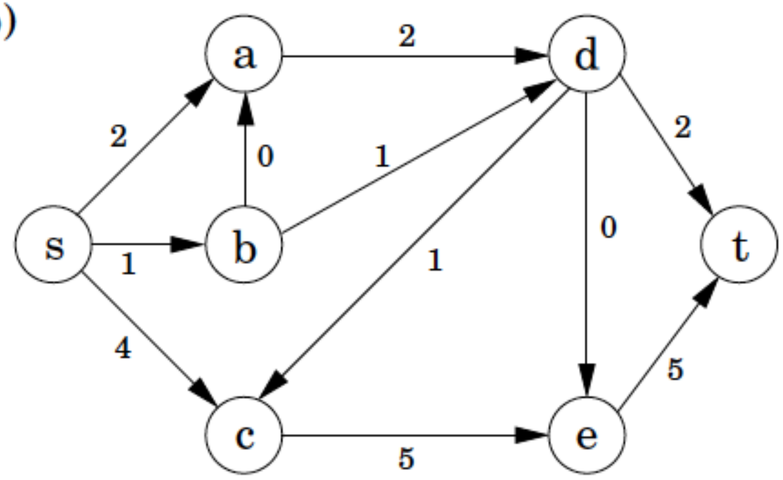
Example:

- Figure (a): network with capacities
- Figure (b): a feasible flow
- In fact, the flow in (b) is optimal (7 units)

(a)



(b)



Flows in Networks

Finding a max flow via Linear Programming:

- Suppose we use a variable f_{uv} for the flow carried by each edge
- Then, the objective function and all the constraints are linear

Objective function: $\sum_{(s,u)\in E} f_{su}$

Constraints

1.Capacity constraints:

$f_{uv} \le c_{uv,}$ for every $(u, v) \in E$

2. Flow preservation:

$$\sum_{(w,u)\in E} f_{wu} = \sum_{(u,v)\in E} f_{uv} \text{ , for every node } u \neq s, t$$

3. Non-negativity constraints:

$f_{uv} \ge 0$, for every $(u, v) \in E$

Flows in Networks

In the example of Figure (a):

max $\quad f_{sa} + f_{sb} + f_{sc}$

s.t.

11 capacity constraints

11 non-negativity constraints

5 flow preservation constraints

27 constraints in total

Solving this => max flow = 7

Note: There are more efficient algorithms for solving max flow (not covered here)

- $O(|V|\ |E|^2)$ [Edmonds, Karp '72]
- $O(|V|^2\ |E|)$ [Goldberg '87]
- $O(|V|\ |E| \log(|V|^2/|E|))$ [Goldberg, Tarjan '86]

Flows in Networks

Recall the max-flow min-cut theorem:

For any graph G = (V, E) with capacities on its edges,

max flow = capacity of minimum s-t cut

In our example, the cut (L, R) shows immediately that the flow of 7 units in Figure (b) is optimal!

The proof of the max-flow min-cut theorem can be done using the LP formulation of the problem (in particular using LP-Duality)

Matching Problems

Types of matching problems that arise in optimization:

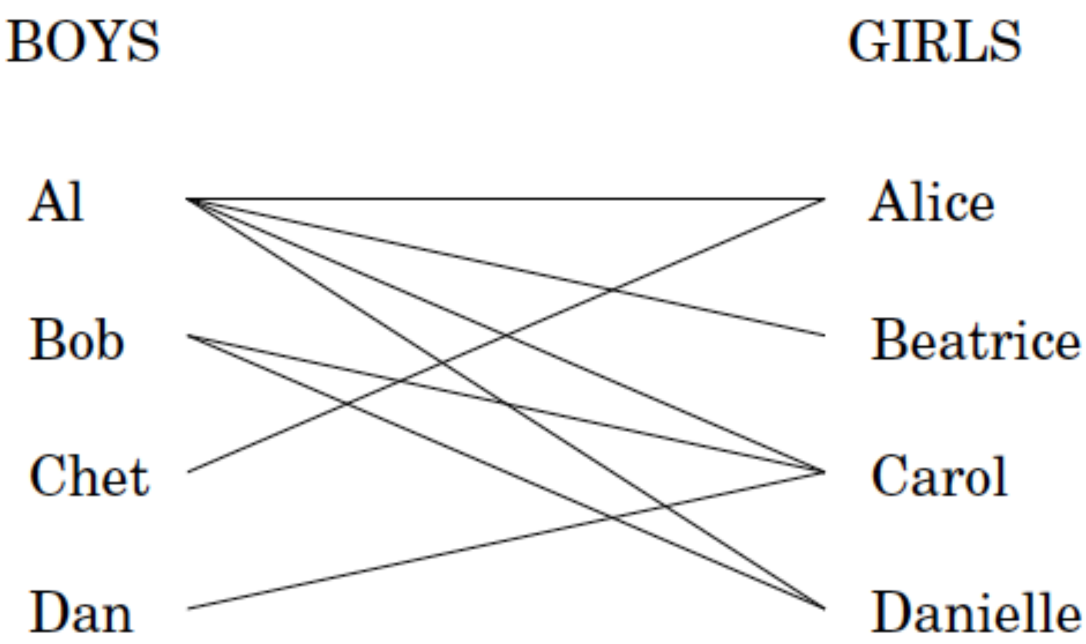
- Maximal matching: find a matching where no more edges can be added
- Maximum matching: find a matching with the maximum possible number of edges
- Perfect matching: find a matching where every vertex is matched (if one exists)
- Maximum weight matching: given a weighted graph, find a matching with maximum possible total weight
- Minimum weight perfect matching: given a weighted graph, find a perfect matching with minimum cost

All the above problems can be solved in polynomial time (several algorithms and publications over the last decades)

Matching in Bipartite Graphs

An interesting special case for matching problems:

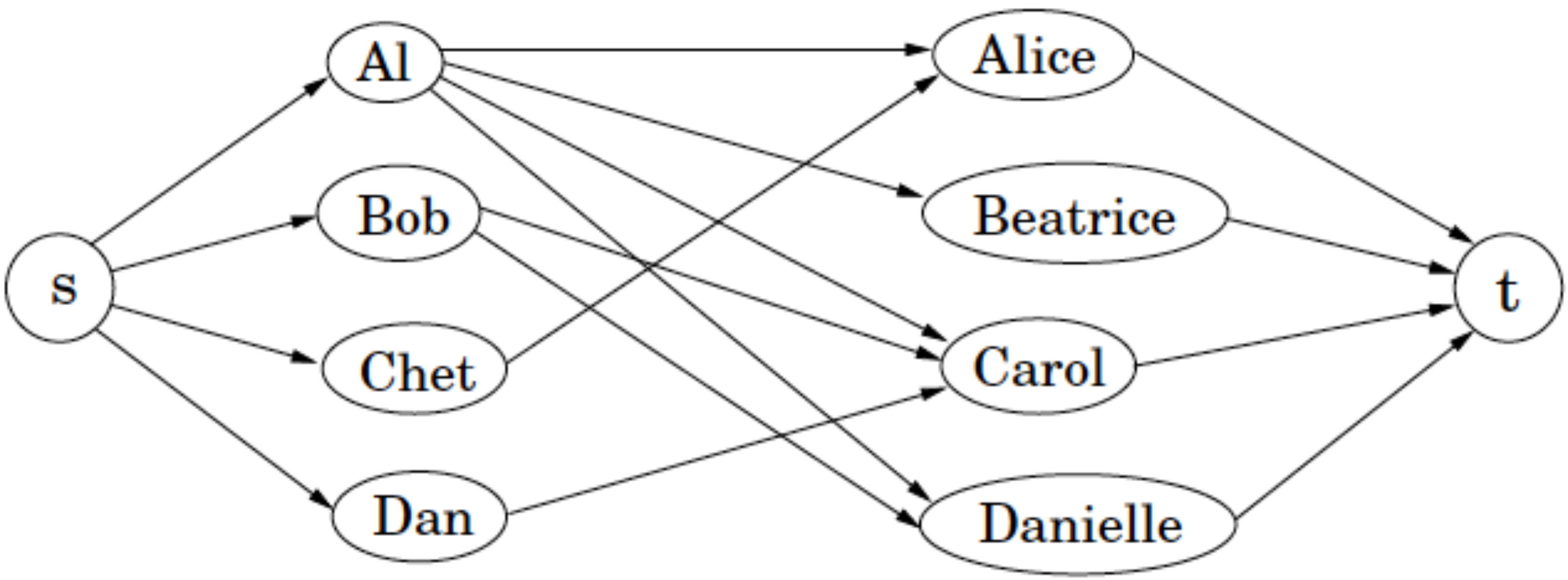
A graph $G = (V, E)$ is called bipartite if V can be partitioned into 2 sets V_1, V_2 such that all edges connect a vertex from V_1 with a vertex from V_2



Q: How can we find a maximum matching in a bipartite graph?

Matching in Bipartite Graphs

We can reduce this to a max-flow problem, and hence to Linear Programming



- Orient all edges from left to right
- Add a source node s, connect it to all of U
- Add a sink node t, connect all of V to t
- Capacities: set them to 1 for all edges

Matching in Bipartite Graphs

Hence:

- a maximum matching for bipartite graphs can be computed in polynomial time
- The graph has a perfect matching if and only if the max flow in the modified graph equals n

Observation: It can be proved that when the capacities are integer numbers, we get an integral flow as an optimal solution, and hence a proper matching as our output

Matching in Bipartite Graphs

An approach without going through flows

- Start with the integer program that describes the matching problem
- Integer programming formulation:
  - Use an integer variable x_e for every edge $e \in E$
  - Let N(v) = edges that come out of node v, the matching should select at most one of them

$$\max \sum_{e \in E} x_e$$

$$\text{s. t.:}$$

$$\sum_{e \in N(v)} x_e \leq 1, \forall v \in V$$

$$x_e \in \{0,1\}, \quad \forall e \in E$$

LP relaxation:

•just set $x_e \geq 0$

•No need to add $x_e \leq 1$, it is implied by the other constraints

Matching in Bipartite Graphs

Constraint matrix of the LP relaxation

- We only have the constraints

$$\sum_{e \in N(v)} x_e \leq 1, \forall v \in V$$

- This is precisely the node-arc incidence matrix for undirected graphs
- Given a node k, and an edge e = (u, v), the entry at row k and column e equals
  - 0, if k ≠ u, k ≠ v
  - 1, if k =u, or k = v

Matching in Bipartite Graphs

Theorem:
For bipartite graphs, the corner points of the polyhedron described by the matching constraints are integral

(proof based on the notion of *total unimodularity*, which is a sufficient condition for integrality of LP solutions)

Corollary: We can compute a maximum matching for bipartite graphs, by solving the LP relaxation

- Recall: the LP algorithms we have discussed identify a corner point optimal solution
- Total unimodularity guarantees that they will return a 0/1 solution

Approximation Algorithms for Vertex Cover and Set Cover

Vertex Cover (VC)

Recall the (optimization) version:

<u>VERTEX COVER (VC)</u>:

I: A graph G = (V,E)

Q: Find a cover C $\subseteq$ V of maximum size, i.e., a set C $\subseteq$ V, s.t. $\forall$ (u, v) $\in$ E, either u $\in$ C or v $\in$ C (or both)

Weighted version:

<u>WEIGHTED VERTEX COVER (WVC)</u>:

I: A graph G = (V,E), and a weight w(u) for every vertex u$\in$V

Q: Find a subset C $\subseteq$ V covering all edges of G, s.t. $W = \sum_{u \in C} w(u)$ is minimized

Many different approximation techniques have been “tested” on vertex cover

Vertex Cover (VC)

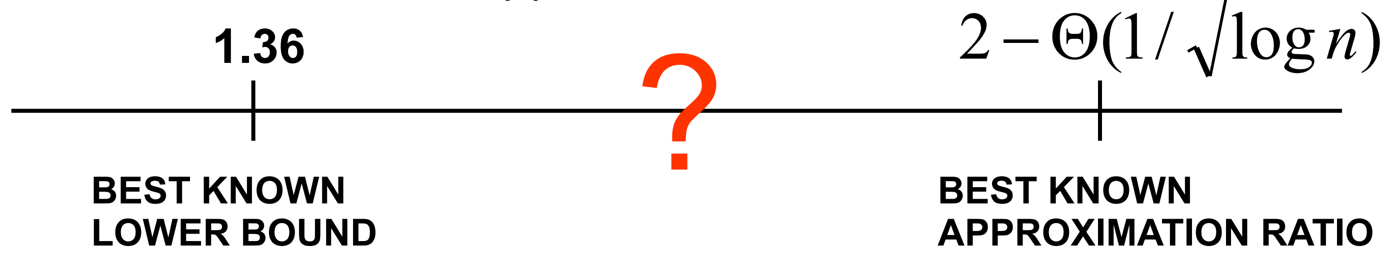
Recall: Greedy-any-edge algorithm (which computes a maximal matching on the input graph) achieves a tight 2-approximation factor and is almost the best known algorithm for VC

Is there a better approximation algorithm ?

We know a lower bound of 1.36 on the approximation factor for VC,

i.e.,

Unless P=NP, VC cannot be approximated with a ratio smaller than 1.36



Big open problem!!

Weighted Vertex Cover (WVC)

- The Greedy-any-edge algorithm does not apply to the weighted case, i.e., a maximal matching does not guarantee anything about the total weight of the solution returned
- Can we have constant approximations here as well?

Recall:

Theorem. The pricing method is 2-approximation for WVC.

Next, we will apply techniques from (Integer) Linear Programming for WVC

Integer Programming Formulations

- Modeling Vertex Cover as an integer program:

<u>Weighted Vertex Cover</u>

$$\begin{aligned} &\min \quad \Sigma_u\, w(u)\, x_u \\ &\text{s.t.} \\ &\qquad x_u + x_v \geq 1 \quad \forall\, (u, v) \in E \\ &\qquad x_u \in \{0,1\} \quad \forall\, u \in V \end{aligned}$$

LP relaxation: Set $x_u \in [0,1]$

Main observation:

- For minimization problems: LP-OPT ≤ IP-OPT (Why?)

Linear Programming Relaxations

- Solving the LP, we get a fractional solution
- But what can we do with it? It is after all not a valid solution for our original problem
- E.g., what is the meaning of having $x_u = 0.8$ for a vertex cover instance?
- LP-rounding: the process of constructing an integral solution to the original problem, given an optimal fractional solution of the corresponding LP
- The process is problem-specific, but there are some general guidelines
- A natural first idea: objects with a high fractional value may be preferred (e.g., if in the LP, $x_u = 0.8$, it may be beneficial to include vertex u in an integral solution)

Linear Programming Relaxations

General scheme for LP rounding:

1. Write down an IP for the problem we want to solve
2. Convert IP to LP
3. Solve LP in O(poly) time to obtain a fractional solution
4. Find a way to convert the fractional solution to an integral one
    - The constructed solution should not lose much in the objective function from LP-OPT
5. Prove that the integral solution has a good approximation guarantee
    - Exploit the main observation to derive bounds with respect to OPT

LP Rounding for WVC

1. First solve:

$$\min \quad \Sigma_u\, w(u)\, x_u$$

s.t.

$$x_u + x_v \geq 1 \quad \forall\, (u, v) \in E$$

$$x_u \in [0,1] \quad \forall\, u \in V$$

2. Let $\{x_v\}_{v \in V}$ be the optimal fractional solution

3. Rounding: Include in the cover all vertices v, for which $x_v \geq ½$

Rationale: Vertices with a high fractional value are more likely to be important for the cover. We also stay "close" in value to LP-OPT

Theorem: The LP rounding algorithm achieves a 2-approximation for the Weighted Vertex Cover problem

Rounding for WVC

Let C be the collection of vertices picked

Claim 1: C is a valid vertex cover

- We started with a feasible LP solution
- Hence, for every edge (u, v), $x_u + x_v \geq 1$
- Thus either $x_u \geq ½$ or $x_v \geq ½$
- By the way we constructed our solution, either u or v belongs to C
- Hence, every edge is covered

Rounding for WVC

Claim2: C achieves a 2-approximation for WVC

Let C be the collection of vertices picked
C corresponds to the integral solution: $y_u = 1$ if $u \in C$, $y_u = 0$ otherwise

Note: $y_u \leq 2\, x_u$, for every $u \in V$

Given this and the main observation:

$$SOL = \sum_{u \in C} w(u) = \sum_{u \in V} w(u) \cdot y_u \leq \sum_{u \in V} w(u) \cdot 2 \cdot x_u = 2 \cdot \text{LP-OPT} \ \leq 2 \cdot OPT$$

Set Cover

SET COVER (SC):

I: a set U of n elements

a family F = {S_1, S_2, …,S_m} of subsets of U

Q: Find a minimum size subset C $\subseteq$ F covering all elements of U, i.e.:

$$\bigcup_{S_i \in C} S_i = U \ \text{ and } |\mathrm{C}| \text{ is minimized}$$

Weighted version:

WEIGHTED SET COVER (WSC):

I: a set U of n elements

a family F = {S_1, S_2, …, S_m} of subsets of U

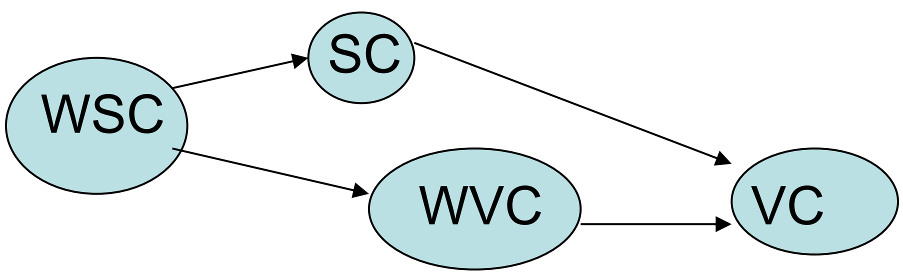
a weight $w(S_i)$ for each set S_i

Q: Find a minimum weight subset C $\subseteq$ F covering all elements of U, i.e.,

$$\bigcup_{S_i \in C} S_i = U \ \text{ and } \mathrm{W} = \sum_{S_i \in C} w(S_i) \text{ is minimized}$$

Set Cover vs Vertex Cover

- (weighted) vertex cover is a special case of (weighted) set cover
- Consider a vertex cover instance on a graph $G = (V, E)$
- Let $U = E$ (i.e., we need to cover the edges)
- One set per vertex, $S_u = \{(u,v) \mid (u,v) \in E\}$, $|F| = |V|$
- In the weighted case, weight of set $S_u = w(u)$



Set Cover vs Vertex Cover

- f_u = frequency of an element $u \in U$ = # of sets S_i that u belongs to

- $f = \max_{u \in U} \{ f_u \}$ = frequency of the most frequent element

- If f=2 (and $w(S_i) = 1$) then (W)SC reduces to (W)VC:
  - $G=(V,E)$, $V= F$, $E= \{ (u,v) \mid S_u \cap S_v \neq 0 \}$

We have seen an approximation algorithm for WSC,
and hence, for SC, WVC and VC:

- Greedy best set is O(log n) (n: the size of the universe U) approximation by a greedy approach

- Next, we will see a LP-based f-approximation for WSC, using an LP rounding approach while extending the 2-approximation for weighted vertex cover

Rounding for WSC

LP relaxation for Set Cover:

$$\min \sum_{S} x_S$$

$$s.t.$$

$$\sum_{u: u \in S} x_S \geq 1, \quad \forall u \in U$$

$$x_S \geq 0, \qquad \forall S \in F$$

Q: How should we round a fractional solution?

Rounding for WSC

LP rounding:

- Solve the LP relaxation
- Fractional solution **x** = $\{x_S\}_{S \in F}$ of cost LP-OPT
- Rounding: if $x_S \geq 1/f$, then include S in the cover

Theorem: The LP Rounding algorithm achieves an approximation ratio of f for the WSC problem

Rounding for WSC

Proof:
Let C be the collection of sets picked

Claim 1: C is a valid set cover

Assume not

- Then there exists some u that is not covered
- => For each set S for which $u \in S$, $x_S < 1/f$
- But then:

$$\sum_{S:u\in S} x_S < \frac{1}{f} |\{S : u \in S\}| = \frac{1}{f} f_u \le \frac{1}{f} f = 1$$

- a contradiction since we found a violated LP constraint

Rounding for WSC

Proof:
Let C be the collection of sets picked

Claim 2: C achieves an f-approximation

Proof very similar to the proof for WVC

Bibliography on Linear Programming

[DPV] S. Dasgupta, C. H. Papadimitriou, U. V. Vazirani : "Algorithms"

Chapter 7, Sections 7.1 – 7.3

Representative exercises: 7.1 – 7.4, 7.6, 7.7, 7.28(a,b), 7.29, 7.30

[Vazirani] V. Vazirani: "Approximation Algorithms"

Chapters: 14,16

Representative exercises: 14.4, 14.7