ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

M.Sc. Program in Data Science
Department of Informatics

Optimization Techniques

Flows, Matchings, and Covering Problems

Instructor: G. ZOIS
georzois@aueb.gr

Outline

- Integrality of LP solutions
  - Total unimodularity
- Flows and Matchings
  - LP formulations
  - Application of total unimodularity
- Vertex Cover and Set Cover
  - Combinatorial algorithms with provable guarantees
  - LP rounding: a technique for deriving approximation algorithms

Integrality of LP Solutions

Question:

- Suppose we have an integer program of the form

$$\max \{ c^T \cdot x \mid A \cdot x \leq b, x \text{ integral} \}$$

- Take the LP relaxation by replacing the integrality constraints with $x_i \geq 0$, for $i = 1,\ldots, n$
- Under what conditions can we guarantee that **all** the corner points of the polyhedron $\{A \cdot x \leq b, x \geq 0\}$ are integral?
- This would enable us to use simplex and solve the initial integer program

Integrality of LP Solutions

Definition:

- Let A be a matrix with entries in {0, -1, +1}
- A is called *totally unimodular* if for every square submatrix T of A, we have det(T) $\in$ {0, -1, +1}

Example:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ -1 & 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Integrality of LP Solutions

Theorem [Hoffman, Kruskal 1956]:

Let A be an integer matrix. The polyhedron $\{A \cdot x \leq b, x \geq 0\}$ has integral corner points for every integral vector b if and only if A is totally unimodular

Can we test if a matrix is totally unimodular?

- Many times, we can argue by just inspecting the matrix
- But having an algorithm remained an open problem for many years
- [Seymour 1980]: the first polynomial time testing algorithm

Total Unimodularity

Some useful properties

Lemma 1: Let A be a matrix with values in {0, -1, +1} and such that every column has at most one +1 and at most one -1.

Then A is totally unimodular

Lemma 2: If A is totally unimodular, then adding a row of the form (0, 0,..., 1, 0,...,0) retains total unimodularity

Lemma 3: If the constraint matrix of a LP is totally unimodular, then the dual LP also has integral optimal corner point solutions

Flows and Matchings

Flows in Networks

(informal) problem statement:

Suppose we want to transport some quantity of a good within a given network, from some source to a destination

The good can be

- Oil to be transported through a network of oil pipes
- Information through a computer network
- Etc

Constraints: each edge in the network has a *capacity*, i.e., the maximum quantity it can carry

- oil pipes have a volume capacity
- A link in a computer network has limits on its bandwidth

Goal: find a way to route the good through the network so as to maximize the total quantity shipped

Flows in Networks

More formally:

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)

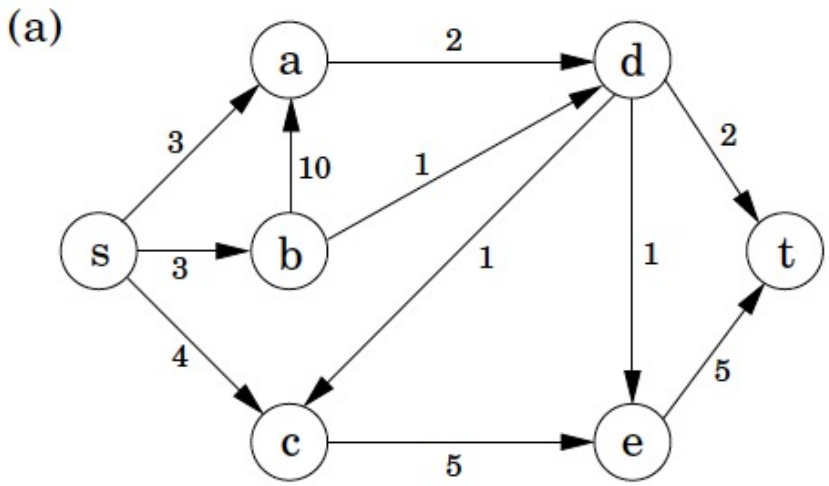
Flow going out of s: $\sum_{(s,u)\in E} f_{su}$

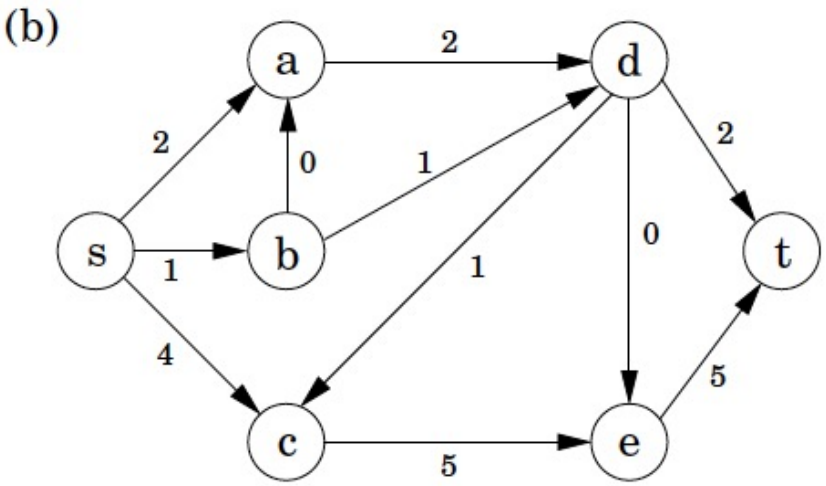
By preservation of flow this equals: $\sum_{(u,t)\in E} f_{ut}$

Flows in Networks

Example:

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





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: $f_{uv} \leq c_{uv,}$ for every $(u, v) \in E$
2. Non-negativity constraints: $f_{uv} \geq 0$, for every $(u, v) \in E$
3. Flow preservation: for every node $u \neq s, t$:

$$\sum_{(w,u)\in E} f_{wu} = \sum_{(u,v)\in E} f_{uv}$$

Flows in Networks

In the example of Figure (a):

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

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

Certificates of optimality:

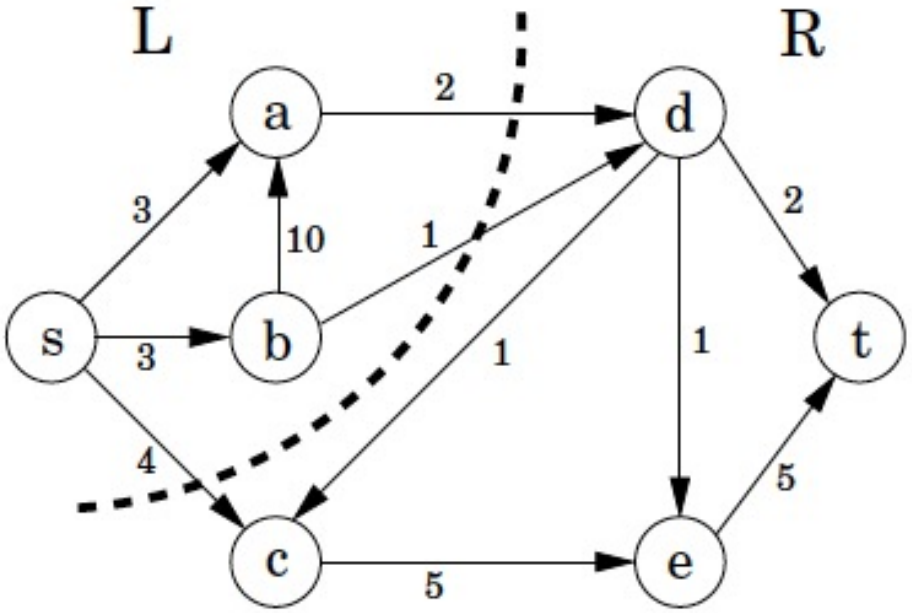
Suppose we have not solved the LP, but we have identified a feasible flow

Can we convince ourselves if it is optimal or not?

Definition: Given a graph G = (V, E), an s-t cut is a partition of the vertices into 2 sets, say L, R, such that $s \in L, t \in R$

Capacity of an s-t cut: sum of capacities of edges crossing the cut in the direction from L to R

Flows in Networks



capacity of cut = 7

Clearly:

max flow ≤ capacity of any s-t cut

(cannot send more flow to t than the capacity of the cut)

Hence:

max flow ≤ capacity of minimum s-t cut

Flows in Networks

In fact we have equality:

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!

Note: One way to prove the max-flow min-cut theorem is by using LP-Duality

Flows in Networks

- Suppose that all the capacity constraints are integers
- Could we then ask for an integral flow?

Theorem:

For any directed graph G = (V, E), the constraint matrix of the max flow LP is totally unimodular

Hence, the optimal solution is attained by an integral flow

Flows in Networks

Sketch of proof

- Let's look at the constraint matrix of the LP
- Need to convert first the equality constraints into ≤-constraints

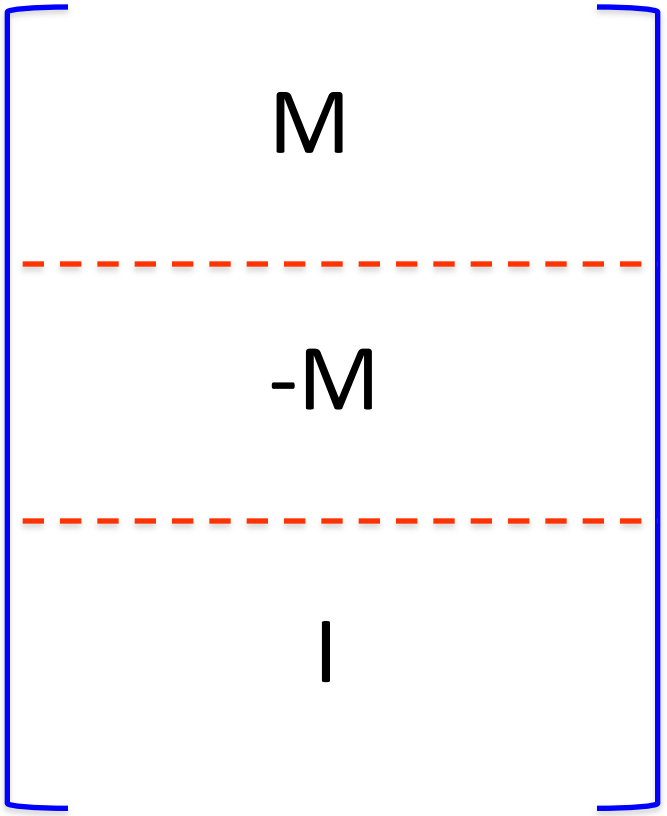
Definition: For a directed graph G, the node-arc incidence matrix is a n x m matrix M where

- n = number of nodes
- m = number of edges
- For an edge e = (u, v), $M_{i,e}$ =
  - 0, if i is not an endpoint of e
  - +1, if i = u (the tail of edge e)
  - -1, if i = v (the head of edge e)

We can write the constraint matrix of our problem in terms of M

Flows in Networks

Sketch of proof



- By Lemma 1, M is totally unimodular
- By Lemma 2, M together with I underneath is also totally unimodular
- With a little more thought, it can be shown that the whole matrix is totally unimodular as well

Matching Problems

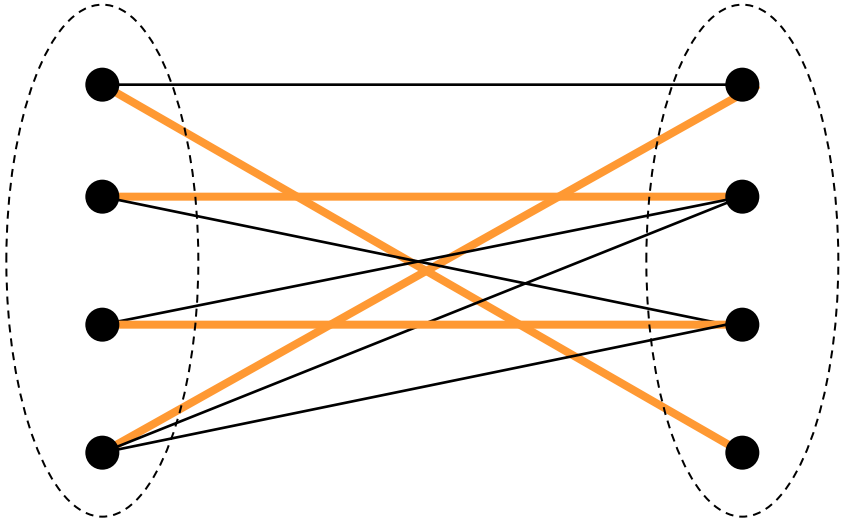
Consider an undirected graph G = (V, E)

Definition: A matching M is a collection of edges $M \subseteq E$, such that no 2 edges share a common vertex

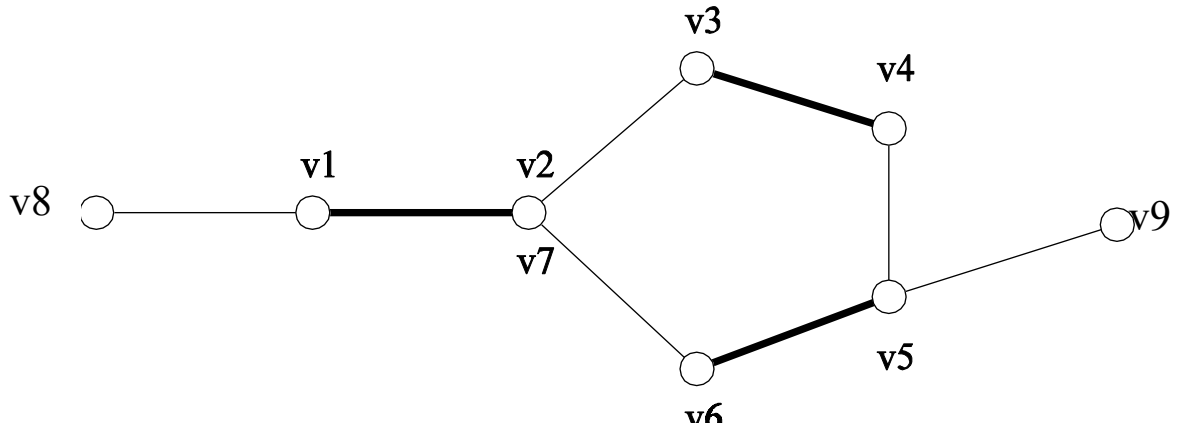
Given a matching M, a vertex u is called *matched* if there exists an edge $e \in M$ such that e has u as one of its endpoints

Matching Problems

Examples



a matching in a bipartite graph



A matching in general graphs (vertex v8 is unmatched)

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 Problems

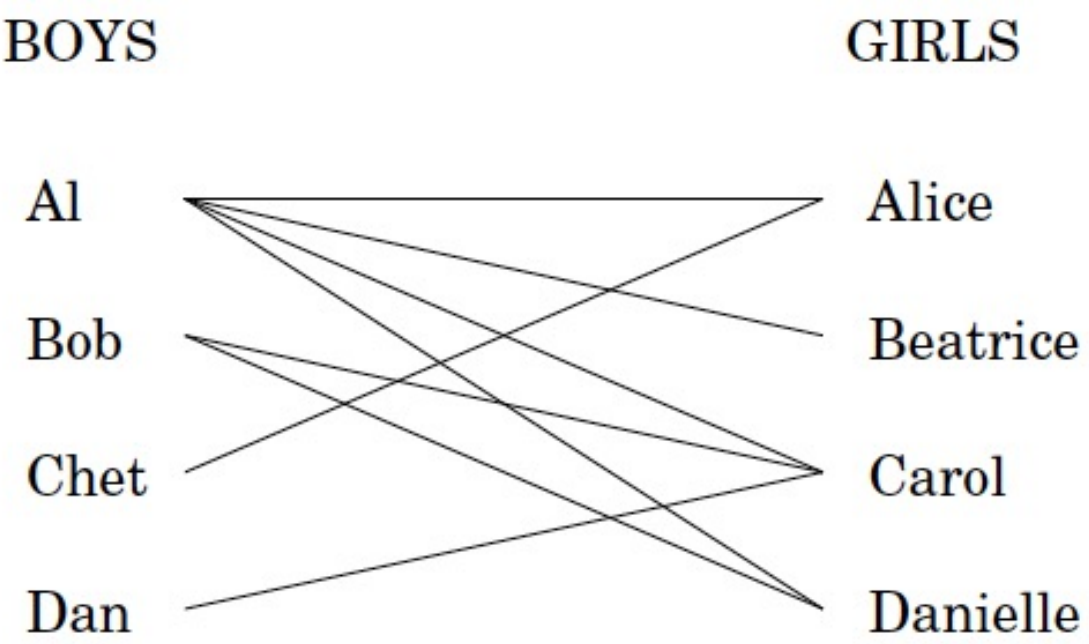
- Trivial algorithm for maximal matching:
  - Start from the empty set of edges
  - Keep adding edges that do not have common endpoints to the current solution
  - Stop when it is not possible to add an edge that does not have any common endpoint with the edges already picked
  - The selected set of edges forms a maximal matching

- More sophisticated algorithms required for maximum matching and perfect matching

Matching in Bipartite Graphs

An interesting special case for matching problems:

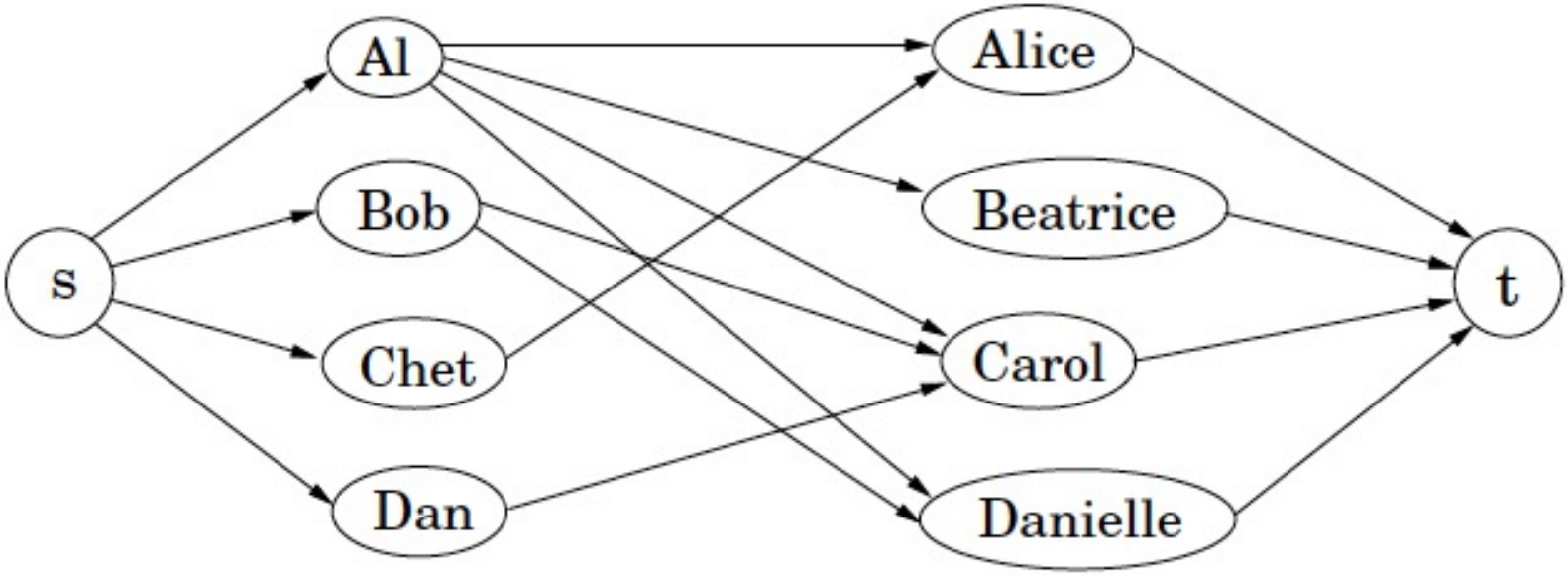
A graph G = (V, E) is called bipartite if V can be partitioned into 2 sets V_1, V_2 such that all edges connect a vertex from V_1 with a vertex from V_2



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

But wait a minute...
What if the max flow we found assigns an outgoing flow of 0.65 to an edge and 0.35 to another edge?

Observation: Because of total unimodularity, we get an integral flow as a 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 $\delta(v)$ = set of edges that have v as one of their endpoints, (the matching should select at most one of them for every node v)

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

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

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

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

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

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

- This yields 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

Theorem:
The node-arc incidence matrix of an undirected graph is totally unimodular if and only if the graph is bipartite (do it as an exercise)

Hence, solving the LP will give us an integer solution, i.e., a maximum matching

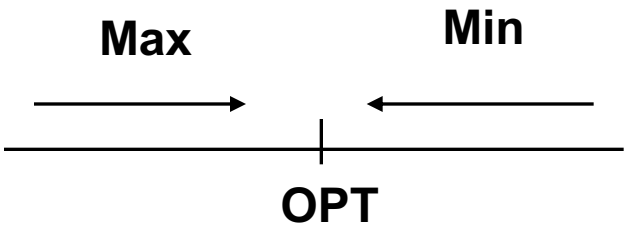
Approximation Algorithms for Vertex Cover and Set Cover

Approximation Algorithms

- Matchings and flows (integral or not) are tractable problems
- 1 call to an LP solver suffices
- What about harder problems (e.g. NP-complete problems)
- Can we still use LP methods to find a solution?
  - We do not expect to always find an optimal solution
  - But we could hope to prove bounds on the approximation quality
- For more on LP-based methodologies for approximation algorithms, see
  - D. Shmoys, D. Williamson. The Design of Approximation Algorithms, Cambridge University Press, 2011

Approximation Algorithms

Recall the definitions from last lecture



Given an instance I of an optimization problem:

- OPT(I) = optimal solution
- C(I) = cost of solution returned by the algorithm under consideration

Definition: An algorithm A, for a minimization problem Π, achieves an approximation factor of ρ (ρ ≥ 1), if for **every** instance I of the problem, A returns a solution with:

$$C(I) \leq \rho\, OPT(I)$$

(analogous definition for maximization problems)

Vertex Cover (VC)

Recall the (optimization) version:

VERTEX COVER (VC):

I: A graph G = (V,E)

Q: Find a cover $C \subseteq V$ of minimum 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:

WEIGHTED VERTEX COVER (WVC):

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 in Bipartite Graphs

Let's start again with this special case

- Take the LP relaxation of maximum matching
- Find the dual linear program
- Make the variables of the dual then to be in {0, 1}

Primal LP

$$\max \sum_{e \in E} x_e$$
$$\text{s. t.:}$$
$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V$$
$$x_e \geq 0, \qquad \forall e \in E$$

Dual LP

$$\min \sum_{v \in V} y_v$$
$$\text{s. t.:}$$
$$y_u + y_v \geq 1, \forall e = (u, v) \in E$$
$$y_v \geq 0, \qquad \forall v \in V$$

The integer version of the dual LP is precisely the vertex cover problem!

Vertex Cover in Bipartite Graphs

An application of LP Duality + Total Unimodularity

By Lemma 3, the dual LP of matching also has integer optimal solutions

Theorem (König):
In a bipartite graph G,
Maximum Matching = Minimum Vertex Cover

- Hence, the problem can be solved efficiently for bipartite graphs (no need for approximation algorithms)
- Equality no longer holds for general, non-bipartite graphs

Vertex Cover (VC)

We will focus first on the unweighted version

Natural greedy algorithms: start picking nodes according to some criterion until all edges are covered

1st approach:

Greedy-any-node

C := ∅ ;

while E ≠ ∅ do

{ choose arbitrarily a vertex u ∈ V;

delete u and its incident edges from G;

Add u to C }

What is the approximation ratio of this algorithm ?

Vertex Cover (VC)

2nd natural approach: start picking nodes and at each step choose the node with the maximum degree

Greedy-best-node

C := ∅ ;

while E ≠ ∅ do

{ choose the vertex u ∈ V with the largest degree; (break ties arbitrarily)

delete u and its incident edges from G;

Add u to C }

Theorem: Greedy-best-node is an O(log n)-approximation algorithm

Vertex Cover (VC)

- The O(logn) ratio of Greedy-best-node is tight
- Can you find an example?

Q: Are there constant factor approximation algorithms?

Vertex Cover (VC)

A different approach:

- To design an approximation algorithm for a minimization problem, we need to find a good lower bound on the optimal solution, for every instance
- We will resort to matching
- Consider an instance of Vertex Cover on a graph G
- Let M be any matching in the graph
- Observation: OPT ≥ |M|
  - The optimal solution needs at least one vertex to cover each of the matched edges
- But we cannot just pick any matching, since it may not be a cover

Matching-based VC

C = ∅;

Find a maximal matching M;

For every (u, v) ∈ M, add both u and v to C

Output C

Vertex Cover (VC)

Theorem: Matching-based VC is a 2-approximation algorithm

Proof:

Claim: The solution returned by the algorithm is a vertex cover

- Suppose not
- Then there is an uncovered edge (u, v)
- But then we could add this edge to the matching M
- Contradiction with the fact that M is a maximal matching

Cost of the solution: $|C| = 2\,|M| \leq 2\,OPT$ (by the observation)

Hence a 2-approximation

Vertex Cover (VC)

A way to implement the maximal matching based algorithm

Greedy-any-edge

C := $\emptyset$;

while E $\neq \emptyset$ do

{ choose arbitrarily an edge (u,v) $\in$ E ;

delete u and v and their incident edges from G;

Add u and v to C; }

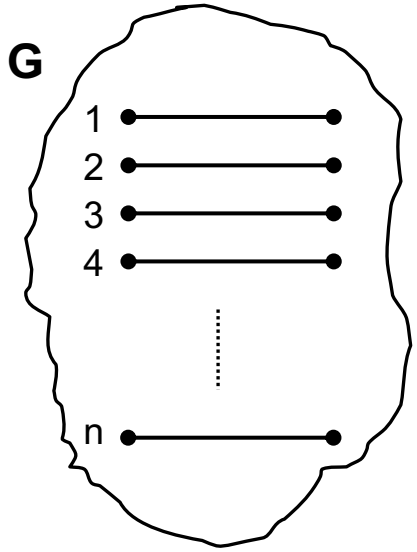
The edges selected by the algorithm form a maximal matching (no 2 edges share a common vertex)

Remark: In contrast to greedy-any-node, greedy-any-edge achieves a constant factor approximation

Vertex Cover (VC)

Tightness of the 2-approximation

Example:



C = 2n

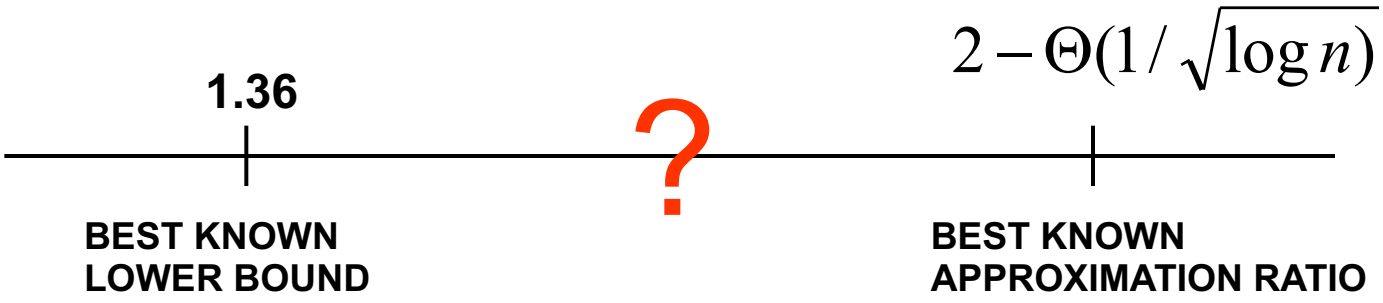
OPT = n

Vertex Cover (VC)

Greedy-any-edge 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 algorithms we have seen so far do not apply to the weighted case
- A maximal matching does not guarantee anything about the total weight of the solution returned
- Can we have constant approximations here as well?
- For this, we will resort to techniques from Linear and Integer Programming

Integer Programming Formulations

- Modeling Vertex Cover as an integer program:

Weighted Vertex Cover

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

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

Recall main observation from last week:

- For minimization problems: LP-OPT ≤ IP-OPT

LP Relaxations and Rounding

- 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)

LP Relaxations and Rounding

General scheme for LP rounding:

1. Write down an IP for the problem we want to solve
2. Convert IP to LP
3. Solve the LP to obtain a fractional solution
   - If the solution is integral, we are done
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.:

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

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

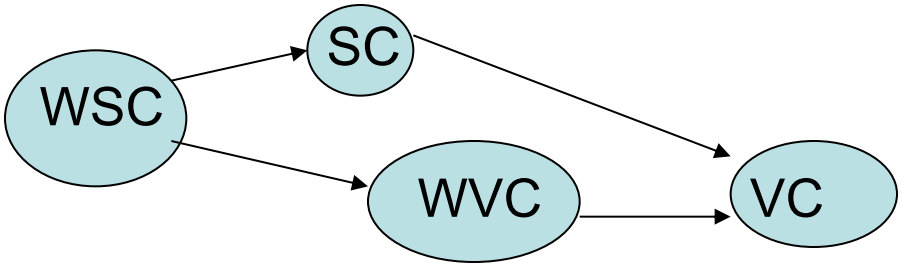
a weight w(S_i) for each set S_i

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

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

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), F = V, U = E
  - We want to cover the edges by nodes
  - S_u is the set of edges covered by node u

There are approximation algorithms for WSC,
and hence, for SC, WVC and VC, of ratios:

- O(log n) (n: the size of the universe U) by a greedy approach
- f, using an LP rounding approach
  - Extending the 2-approximation for weighted vertex cover

Weighted Set Cover (WSC)

In a similar spirit as for Vertex Cover:

Greedy-best-set

C := $\emptyset$;

while C $\neq$ U do

{ choose the best set S;

remove S from F;

C := C U S ; }

C: elements covered before iteration i

S: Set chosen at iteration i

Q: What does "best set" mean ?

S covers |S-C| new elements

Covering those elements costs *w(S)*

Every element x $\in$ S essentially costs $\frac{w(S)}{|S-C|} = p(x)$ = "cost-effectiveness" of S

Best set: the set with the smallest cost-effectiveness

Weighted Set Cover (WSC)

Approximation analysis of Greedy-best-set

Let $x_1, x_2, \ldots, x_k, \ldots, x_n$ be the order in which the elements of U are covered

$S_1, S_2, \ldots S_i, \ldots$ be the order in which sets are chosen by the algorithm

Suppose set S_i covers element x_k

Claim: $p(x_k) \le \dfrac{OPT}{n-k+1}$

$C = \bigcup_{j=1}^{i-1} S_j$ elements covered by iterations 1,2,…,i-1

- U-C: uncovered elements before iteration i
- |U-C| ≥ n-k+1, since element x_k is covered in iteration i

Weighted Set Cover (WSC)

- These elements of U-C are covered in the optimal solution by some sets at a cost of at most OPT
- Among them there must be one set with cost-effectiveness at most

$$\leq \frac{OPT}{|U-C|} \leq \frac{OPT}{n-k+1}$$

- the set S_i was picked by the algorithm as the set with the best cost-effectiveness at that moment (and it covered x_k)

- that is $p(x_k) \leq \frac{OPT}{n-k+1}$

$$W = \sum_{k=1}^{n} p(x_k) \leq \sum_{k=1}^{n} \frac{OPT}{n-k+1} = OPT \sum_{i=1}^{n} \frac{1}{k} = OPT \cdot H_n = O(\log n) OPT$$

LP Rounding for WSC

LP relaxation of 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 $\mathbf{x} = \{x_S\}_{s \in F}$ 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:

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

- 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