Graph problems

Vertex Cover, Dominating set,
Clique, Independent set

This lecture

- Several basic graph problems:
  - Finding subsets of vertices or edges with simple property – as small or as large as possible
  - Relations

1

Stating the Wireless Network Problems

Some Wireless Network Problems

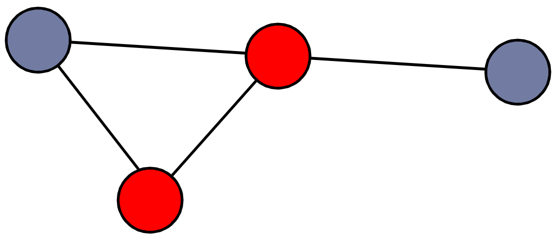
- How select towers and block all wireless communication?
- How to disseminate information in a wireless network?
- How to achieve network maximum capacity?
- How to select wireless MAC scheduling?

2

Stating the Graph problems

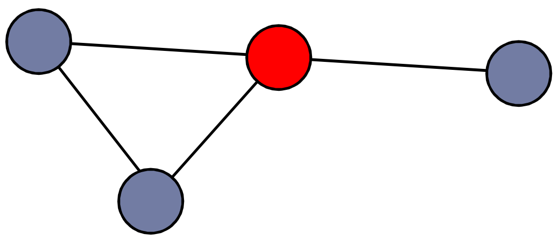
Vertex cover

- Set of vertices $W \subseteq V$ with for all $\{x,y\} \in E$: $x \in W$ or $y \in W$.
- Vertex Cover problem:
  - Given G, find vertex cover of minimum size



Dominating Set

- Set of vertices W such that for all $v \in V$: $v \in W$ or v has a neighbor w with $w \in W$.
- Dominating Set problem: find dominating set of minimum size
- E.g.: facility location problems



Clique and Independent Set

- Clique: Set of vertices W with for all $v, w \in$ W: $\{v,w\} \in$ E.
- Independent set: Set of vertices W with for all $v, w \in$ W: $\{v,w\} \notin$ E.
- Clique problem: find largest clique.
- Independent set problem: find largest independent set

Relations

- The following are equivalent
  - G has an independent set with at least k vertices
  - The complement of G has a clique with at least k vertices
  - G has a vertex cover with at most n-k vertices

NP-complete

- Each of the following problems is NP-complete:
  - Decide if given graph G has vertex cover of size at most a given integer K
  - Decide if given graph G has dominating set of size at most a given integer K
  - Decide if given graph G has independent set of size at least a given integer K
  - Decide if given graph G has clique of size at least a given integer K
- Proofs omitted in this lecture

3

On approximation 1:
Clique and Independent Set

Approximation

- Clique, IS are very hard to approximate
- Ratio 2 approximation algorithm for Vertex Cover
  - H = G, S = $\emptyset$
  - **repeat until** H is empty
    - Select an edge $\{v,w\} \in E$. Put *v* and *w* in S, and remove *v*, *w*, and all adjacent edges from H
- Proof of ratio: for each selected two vertices, at least one belongs to optimal solution

Warning

- The vertex cover approximation does NOT give a ratio for an approximation for Clique or Independent Set
  - E.g., take a graph with 1000 vertices and a vertex cover of size 498.

4

Bipartite graphs

Vertex cover in bipartite graphs

- In a bipartite graph: the minimum size of a vertex cover equals the maximum size of a matching
  - If we have a matching: each edge must have an endpoint in the vertex cover
  - In the other direction:
    - Let X be a vertex cover. Take flow model. Look at the edges from *s* to vertices in X, and from vertices in X to *t*. These form a cut. Now use the min-cut max-flow theorem.

Bipartite graphs

- We saw: min vertex cover in bipartite graph equals maximum matching
- Corollary: Independent Set in bipartite graphs is polynomial time computable

Special cases for Independent Set

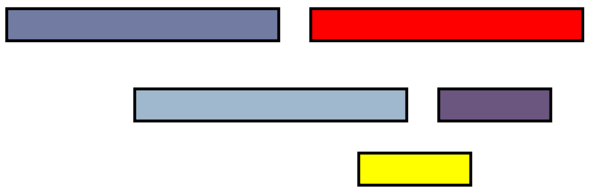
- For many types of graphs (e.g., planar) IS stays NP-complete when restricted to these graphs.
- Some special cases are polynomial
  - Trees, bounded treewidth
  - Bipartite graphs
  - Interval graphs, circular arc graphs

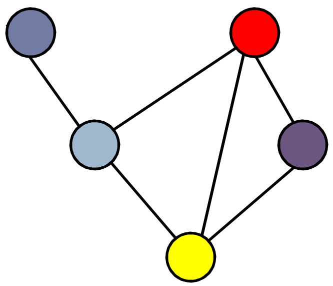
5

Interval graphs and generalizations

Interval graphs

- Each vertex represents an interval
- Edge between two vertices when their intervals have non-empty intersection
- Independent set on interval graphs models room-booking problem





Greedy algorithms on interval graphs

- Greedy algorithm for IS on interval graphs is optimal:
  - Repeat:
    - Select vertex *v* with leftmost right-endpoint of interval
    - Remove *v* and its neighbors
- Clique is trivial on interval graphs

Circular arc graphs

- Like interval graphs, but now `intervals on a circle' (model of repeating time…)
- If we remove a vertex and its neighbors from an circular arc graph, then we get an interval graph
- Algorithm, e.g.:
  - **for** all vertices *v* **do**
    - Compute the maximum independent set of G minus *v* and its neighbors
  - If the largest i.s. is found for *v**, output that set with *v** added.