# Pursuit Evasion

as presented in Megiddo et al. 1988:
„The complexity of searching a graph“

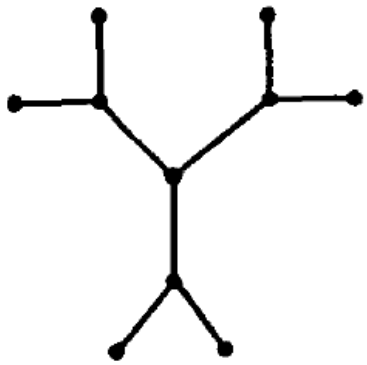
Sebastian Blohm

## What is the Problem?

- Basically: Find out, how many searchers it takes to catch a fugitive (e.g. **pacman**), if he moves infinitely fast.
- Graph model: pacman can walk along edges and swap to all its neighbours. He gets caught, if a searcher moves along the edge he is on.
- Alternative model: clearing a network of pipes from toxic gas moving filters along the pipes.



## What is the Problem? (example)





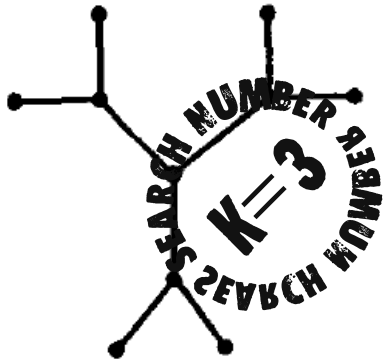
## What is the Problem? III

- Given a particular graph G the **search number s(G)** is the number of searchers needed to find the fugitive, filter the gas etc.
- Graphs can be classified by an integer k, s.t. **s(G) = k.**
- **How can we do that efficiently?**



## What is the Problem? (example)

- This graph can be cleared by three searchers:






## Why is this interesting?

- Papers have been published using this in:
  - » defense strategies
  - » graph layout
  - » search for information on the web
- The common point is that the search number is a **good measure for complexity**.
- What I am interested in: Is this maybe a measure for how much computational effort a human planner has to put into a task?



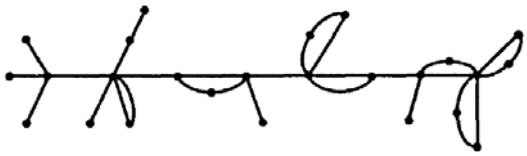
## Outline of Talk

- what is the problem? (done)
- questions up to now
- sample questions
- taking minors
- characterizing graphs with k=2
- characterizing graphs with k=3
- the general case in NP-complete
- for trees it is easier to solve
- applications
- summary



## Sample Questions

- What is the search number of this graph?



- Show that $F_{s(G)=k}$ is closed under taking minors.
- Why is it easier to limit the search number of trees?
- Show that an algorithm can determine if a graph is 3-searchable in linear time (with appropriate hints).

## ...closed under taking minors

- remember from last class or Vida's lecture the notion of a minor-closed class.
- we can define the class $F_{s(G)=k}$ Containing all graphs with search number s(G)= k.
- To see that $F_{s(G)=k}$ is closed under taking minors, we would have to check, if the operations involved in minoring can increase the search number:
  - **remove** vertices or edges
  - **contract** edges
  - **reduction**, special case of contraction removing all vertices of degree 2.
- No, **these operations can only decrease s(G)**

$$G \preceq H \Leftrightarrow s(G) \leq s(H)$$

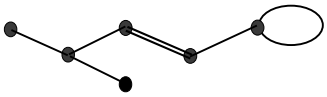
- according to the Graph Minors Theorem, we can now find a finite obstruction set.



## characterizing graphs with k=2

- THEOREM 5: for any reduced graph G the following are equivalent:
  a) s(G) = 2
  b) G **does not contain** any of these graphs (as a minor):

  c) G **consists of a path** (with double edges) with individual edges and loops attached to it:

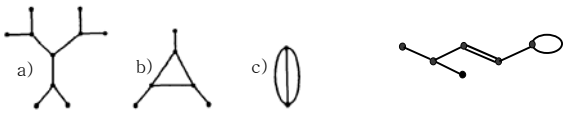


- Why is that nice? **– We have closed our obstruction set.**



## characterizing graphs with k=2

- b) -> c): If G **does not contain** any of these graphs, then it **consists of a path** like this.

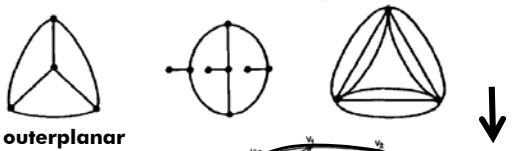



- look at minor b: every biconnected component which more than two verts of degree > 2 would contain it.
  → no not-reducible circles → G is a tree (with self-loops and parallel edges)
- the forbidden minor a) prevents G (once beeing a tree) from being anything else then a path. (Verify by trying to remove any edge without making it satisfy c)
- c) excluded triple edges
- a), b) and c) form a complete obstruction set

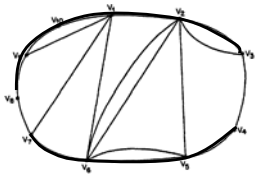


## characterizing graphs with k=3

- LEMMA 7: for any reduced, biconnected graph G the following are equivalent:
  a) s(G) = 3
  b) G **does not contain** any of these graphs (as a minor):



  c) G is **outerplanar** and **bipolar**.





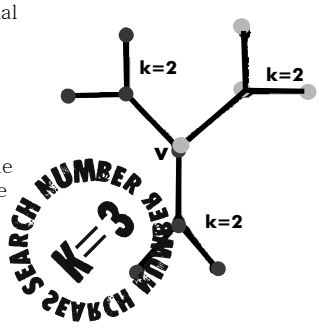
## the general case is NP-complete

- Deciding, weather s(G) = k by trying out gets **exponentially** harder because:
  a) you can start searching at any vertex
  b) you might have to continue from any vertex with every adjacent edge
  c) there might be some strategy involving backing up. (disproven)
- If you get a suggested search plan (which will be polynomial (in fact linear) to |G|), you just have to try it out to verify -> **polynomial in non-deterministic case**.
- (**NP-Completeness** can be proven by reduction from: „MIN-CUT INTO EQUAL SIZED SUBSETS.)



## finding k for trees

- For trees, k can be determined **recursively**:
- PARSONS' LEMMA. For any tree T and integer k = 1, **s(T) = k+1** if and only if T has a vertex v at which there are three or more **branches that have search number k** or more.
- So what is a **branch**?
  A branch at v is a maximal subtree in which v has degree 1.
- Our example has 3 branches at v:
- An algorithm to determine k thus does **BFS** on a tree and then **counts from the leafs up**.






## Applications

- P. Spirakis , B. Tampakas, Distributed pursuit-evasion: some aspects of **privacy and security in distributed computing**, (1994)
  Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing, p.403, Los Angeles
  http://citeseer.nj.nec.com/spirakis94distributed.pdf
- K. Skodinis, Computing **optimal linear layouts** of trees in linear time (1999)
  European Symposium on Algorithms
  http://citeseer.nj.nec.com/317941.html



## Summary

- It is interesting to find the search number of a graph.
- The class $F_{s(G)=k}$ is closed under taking minors.
- There are three forbidden minors to determine $F_{s(G)=2}$ and $F_{s(G)=3}$
- The general case in NP-complete
- For trees it can be solved recursively.
- There are some nice applications.

**References:**

Megiddo et al. : The complexity of searching a graph
http://doi.acm.org/10.1145/42267.42268

this presentation: http://www-lehre.inf.uos.de/~sblohm/