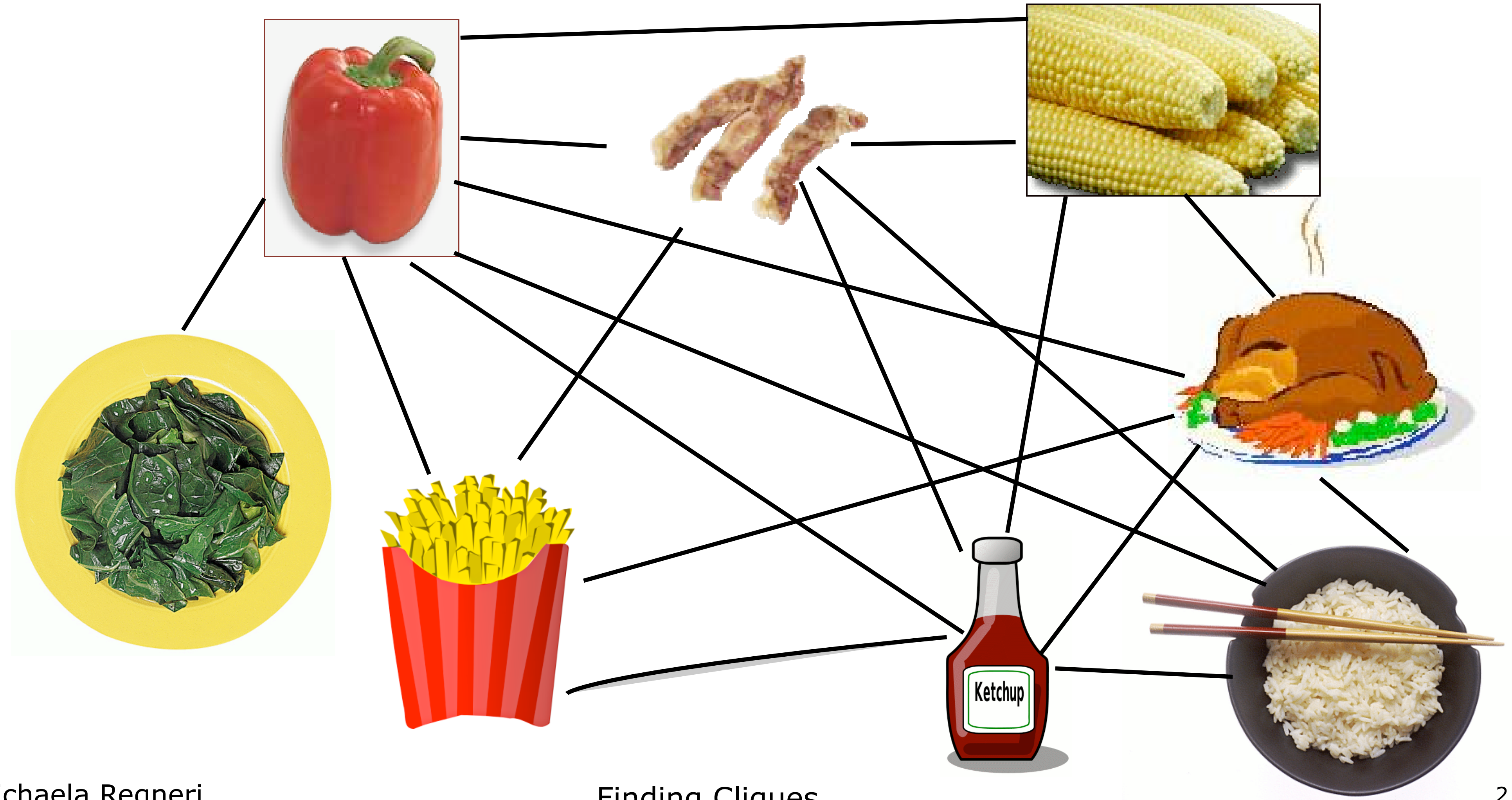
Finding All Cliques of an Undirected Graph

Seminar „Current Trends in IE“ WS 06/07

Michaela Regneri
11.01.2007

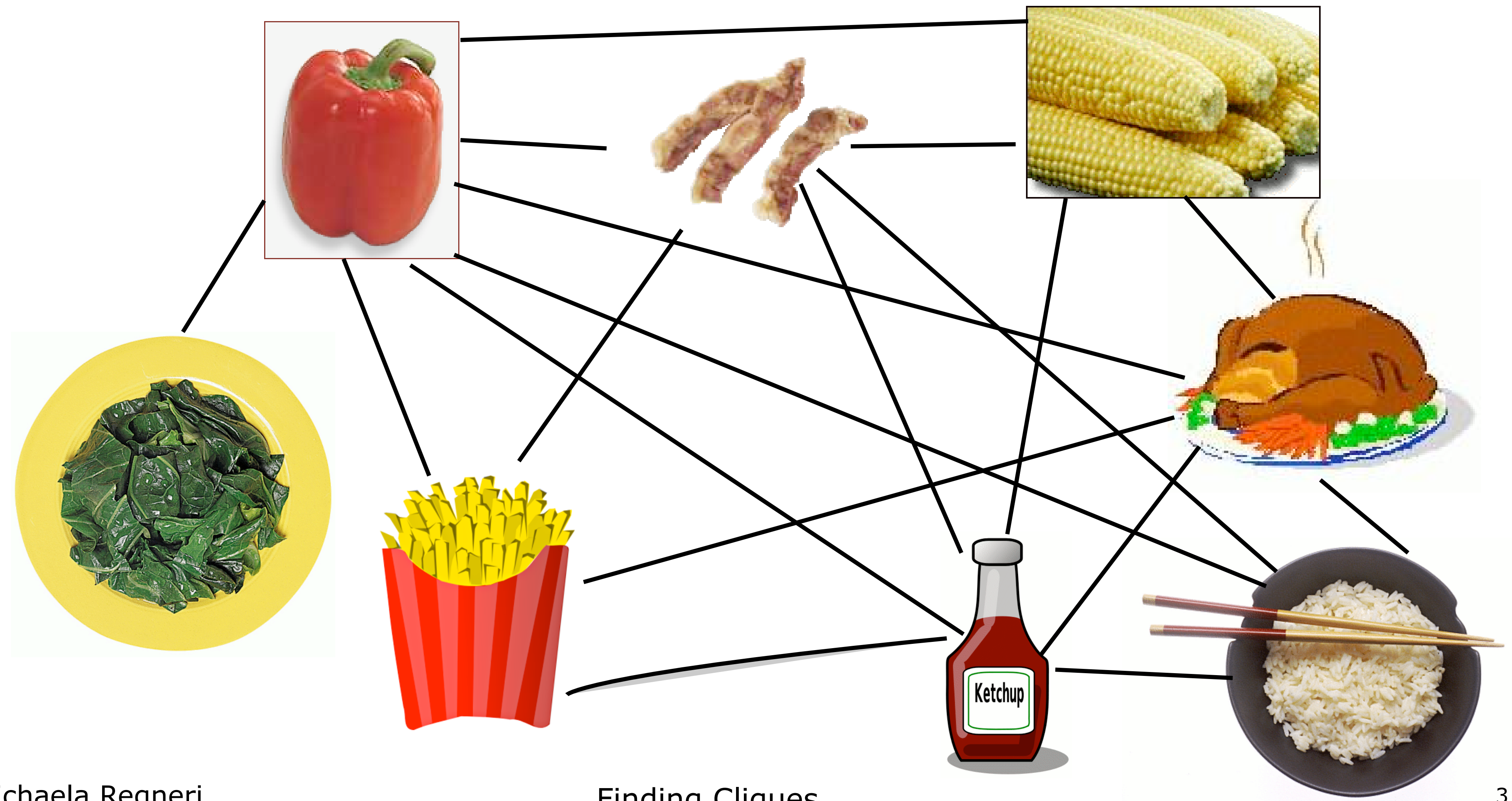
Motivation

- How to put as much left-over stuff as possible in a tasty meal before everything will go off?



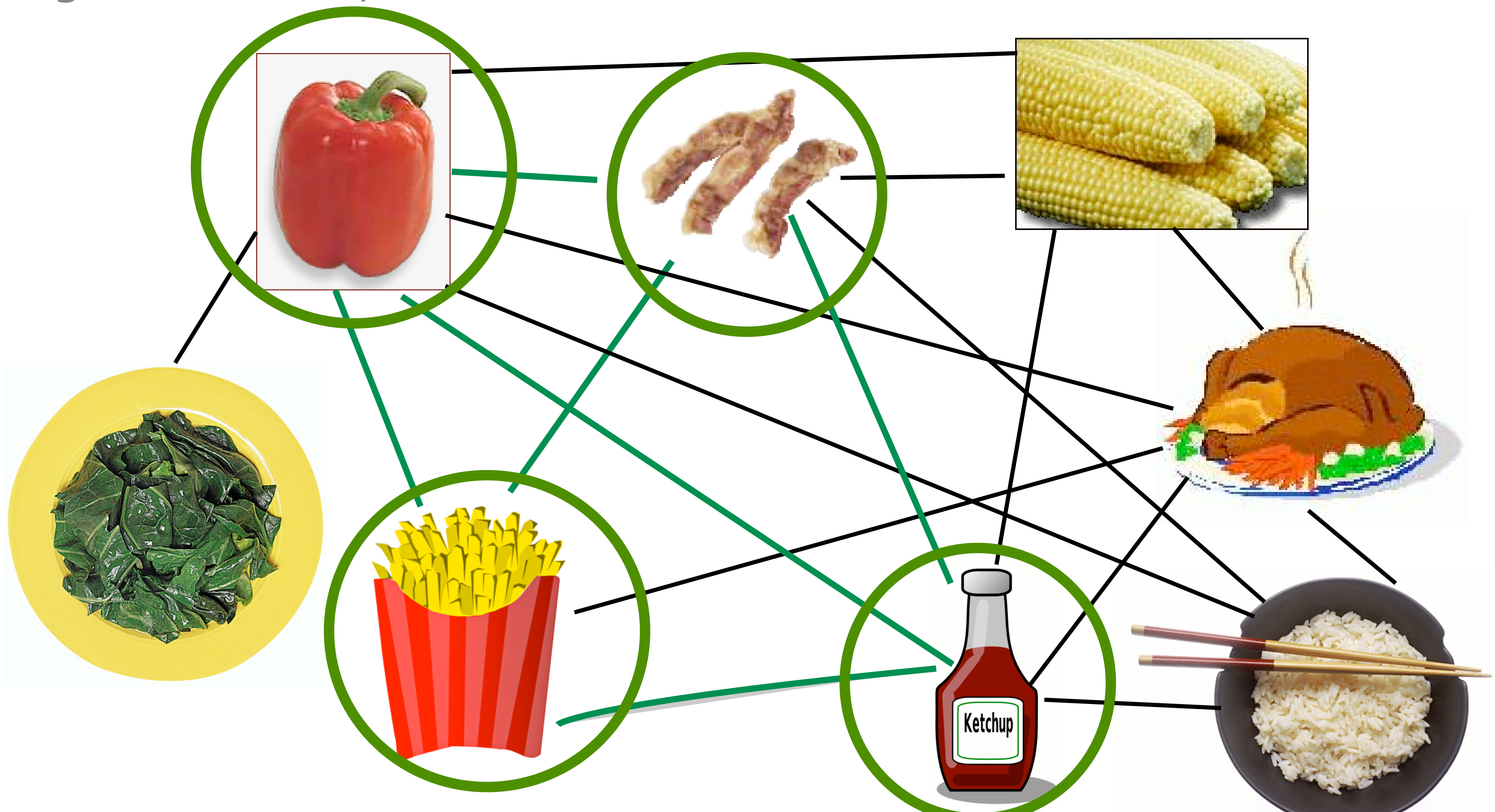
Motivation

- Find the largest collection of food where everything goes together! Here, we have the choice:



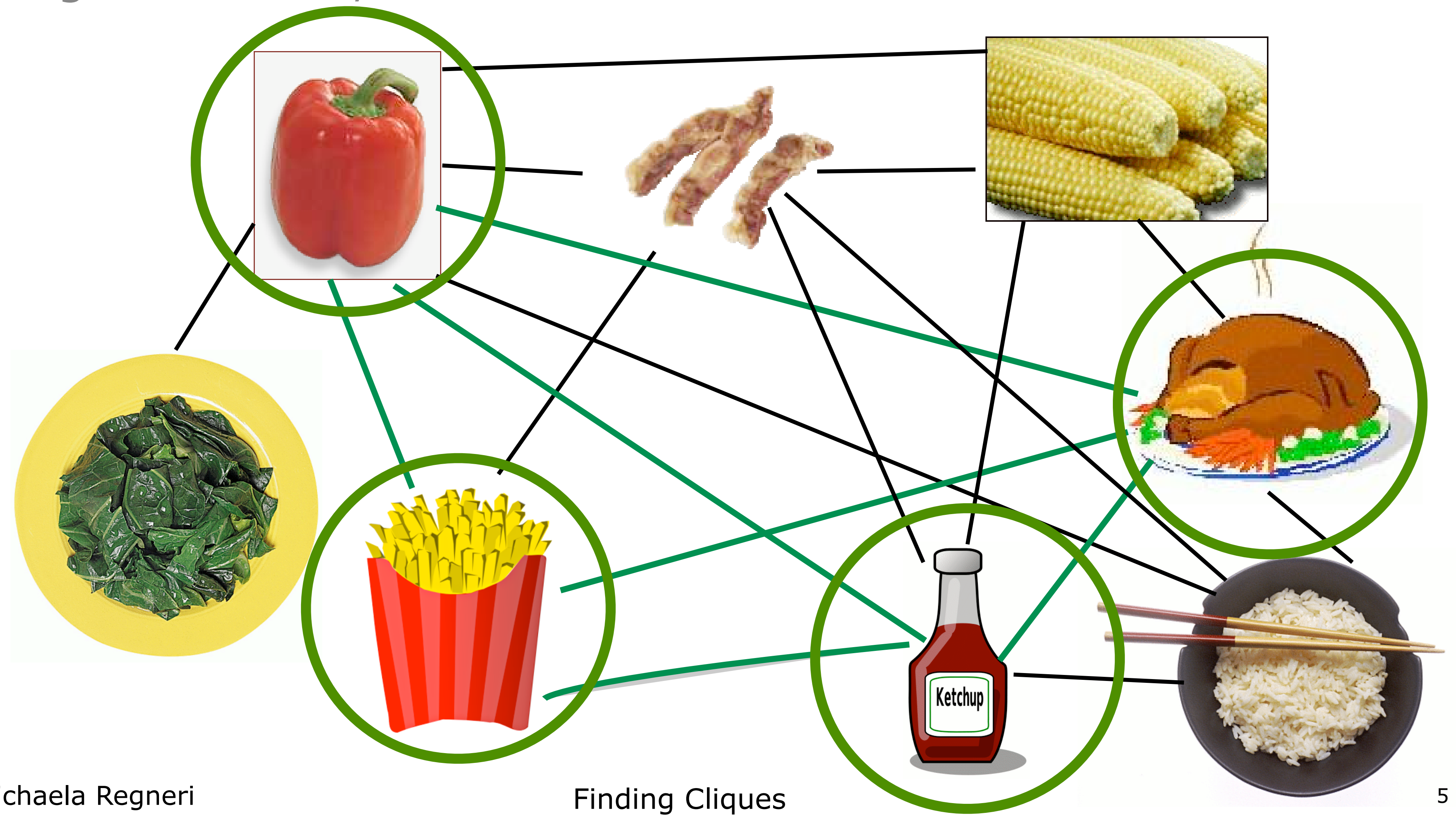
Motivation

- Find the largest collection of food where everything goes together! Here, we have the choice:



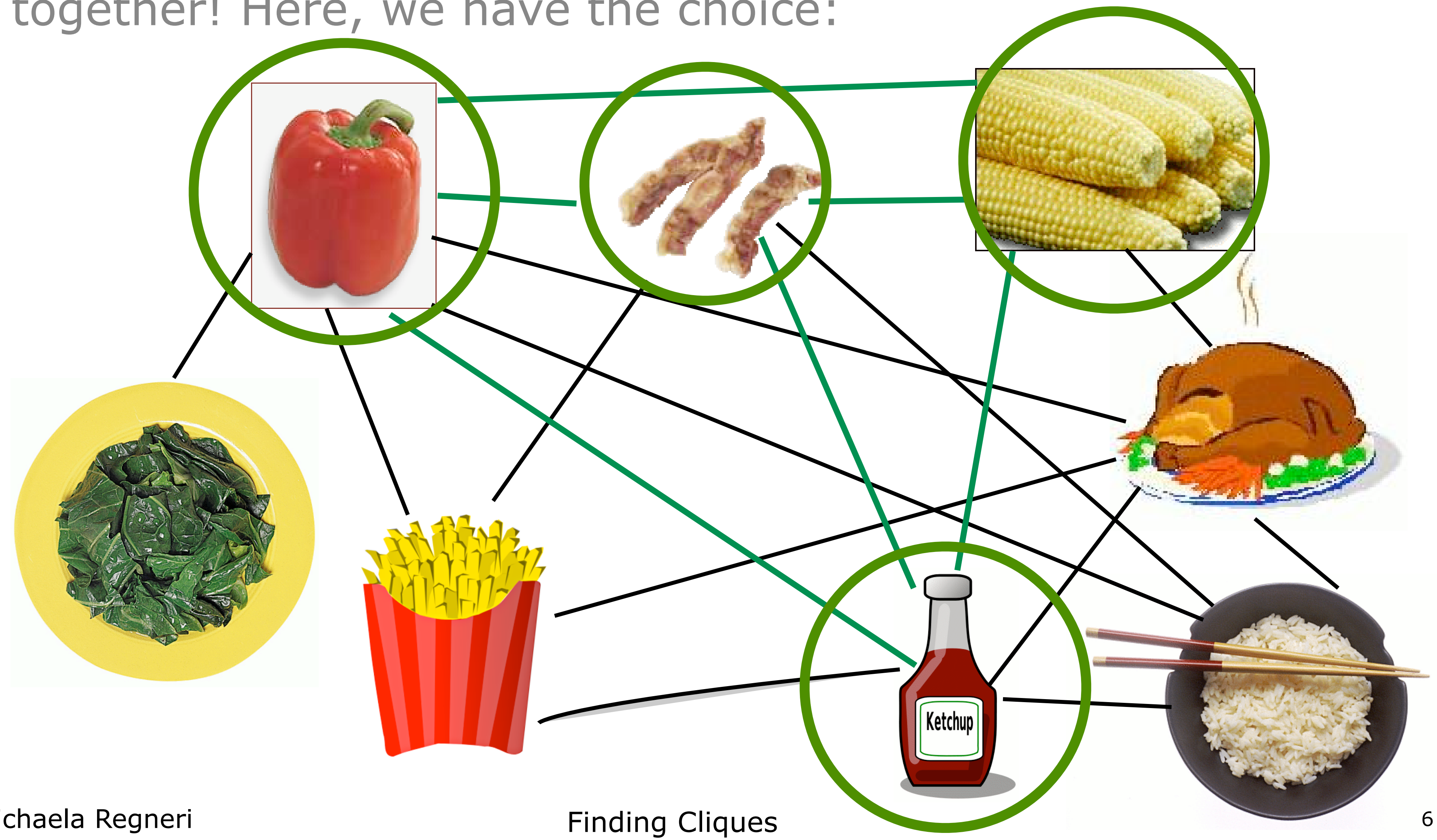
Motivation

- Find the largest collection of food where everything goes together! Here, we have the choice:



Motivation

- Find the largest collection of food where everything goes together! Here, we have the choice:



Outline

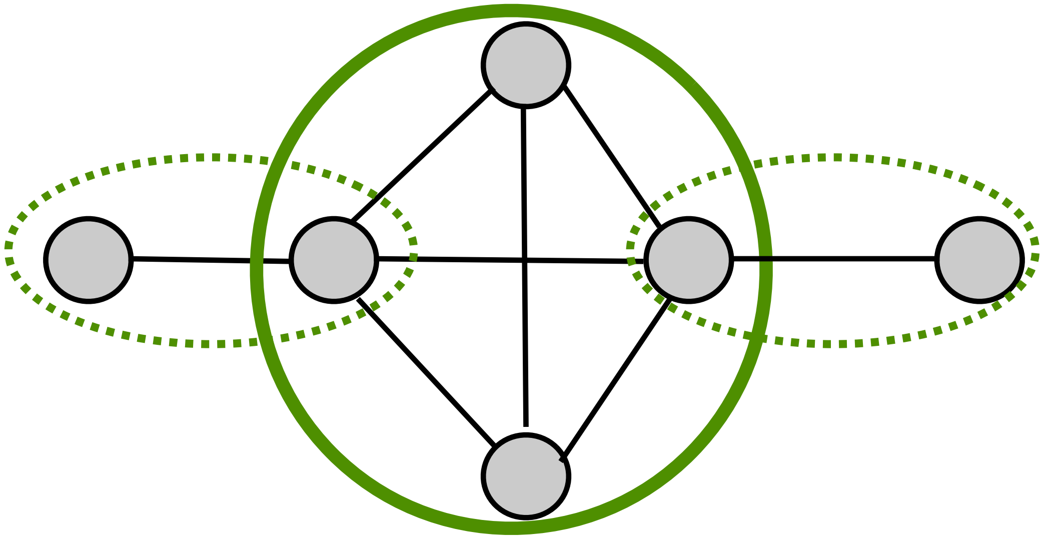
- Introduction to cliques
- The Bron-Kerbosch algorithm
- Applications
- Conclusion

Outline

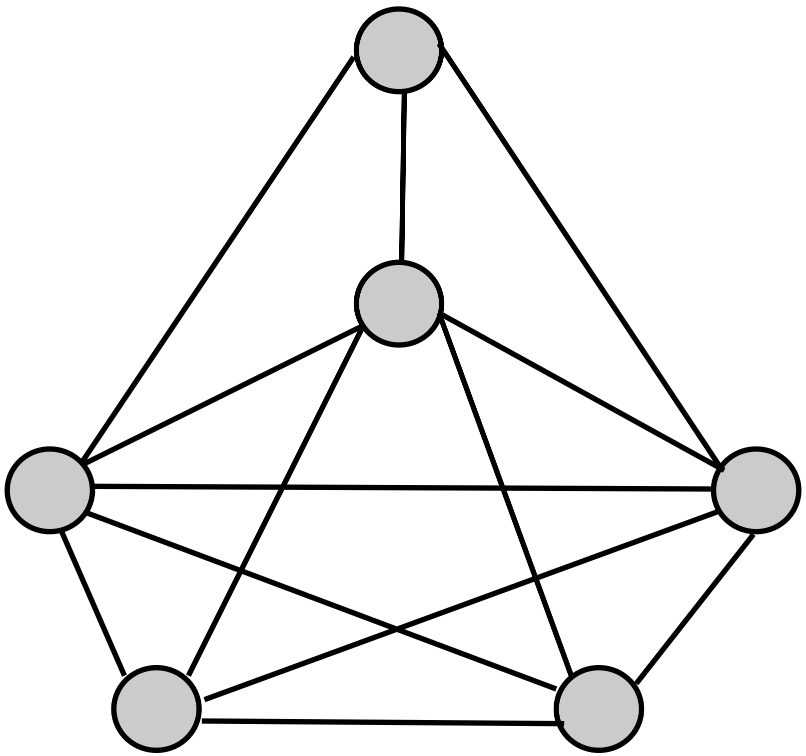
- Introduction to cliques
- The Bron-Kerbosch algorithm
- Applications
- Conclusion

Cliques (according to Bron and Kerbosch 1973)

- complete subgraph of a graph: part of a graph in which all nodes are connected to each other
- cliques: maximal complete subgraphs (not subsumed by any other complete subgraph)

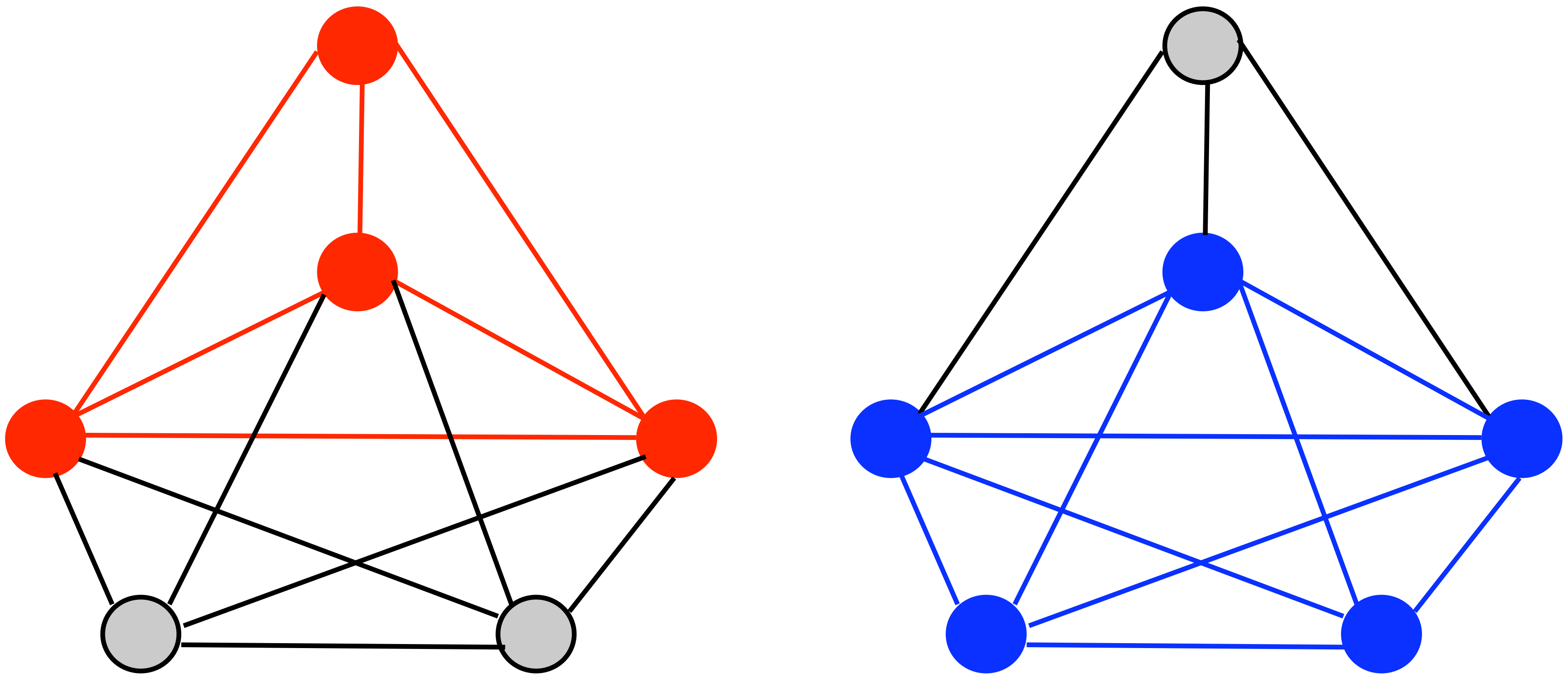


A graph



How many cliques?

The cliques



How can we find them efficiently?

Outline

- Introduction to cliques
- The Bron-Kerbosch algorithm
- Applications
- Conclusion

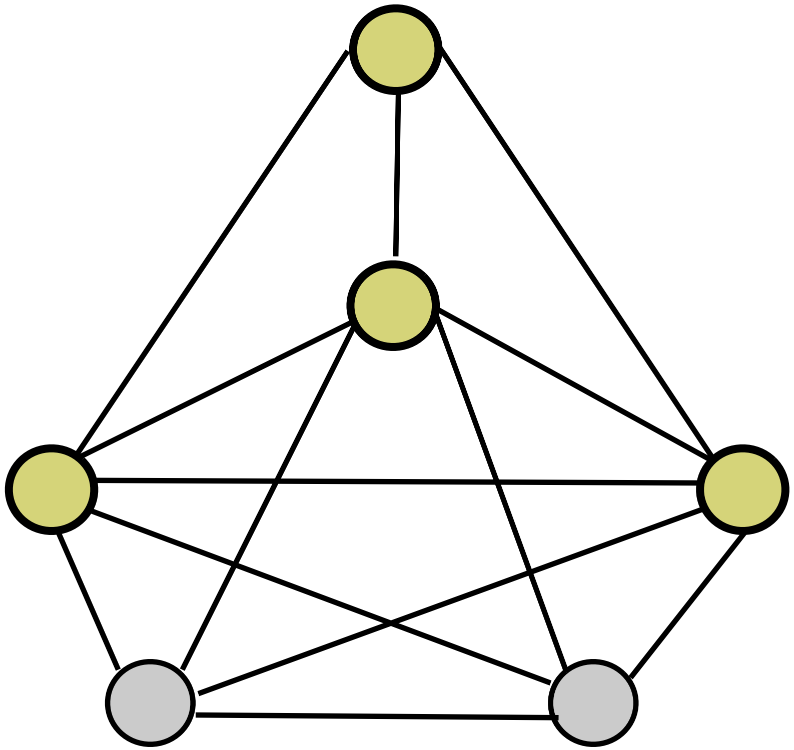
The Bron-Kerbosch algorithm

- finding all cliques is expensive
- the number of cliques can grow exponentially with every node added
- Bron and Kerbosch (1973):
  - An algorithm to compute all cliques in linear time (relative to the number of cliques)
  - still widely used and referred to as one of the fastest algorithms (cf. Stix 2004)

Different sets and types of nodes

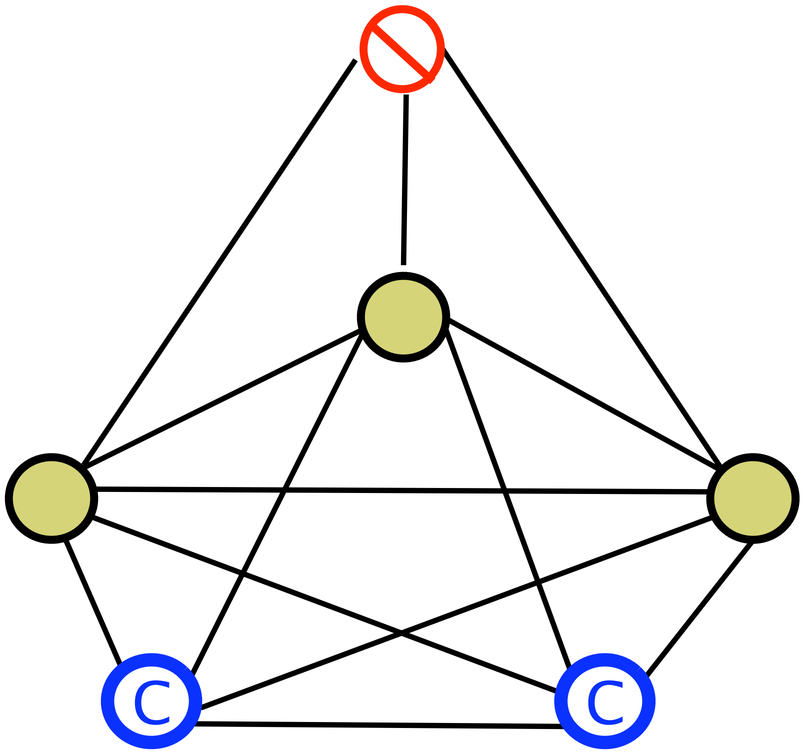
- the nodes already defined as part of the clique (*compsub*) (initially empty)
- the *candidates*, connected with all nodes of *compsub*
- *not*, the nodes already processed which lead to a valid extensions for *compsub* and which shouldn‘t be touched
- a selected candidate
- nodes which are not considered in the current step

A clique is found if (and only if)...



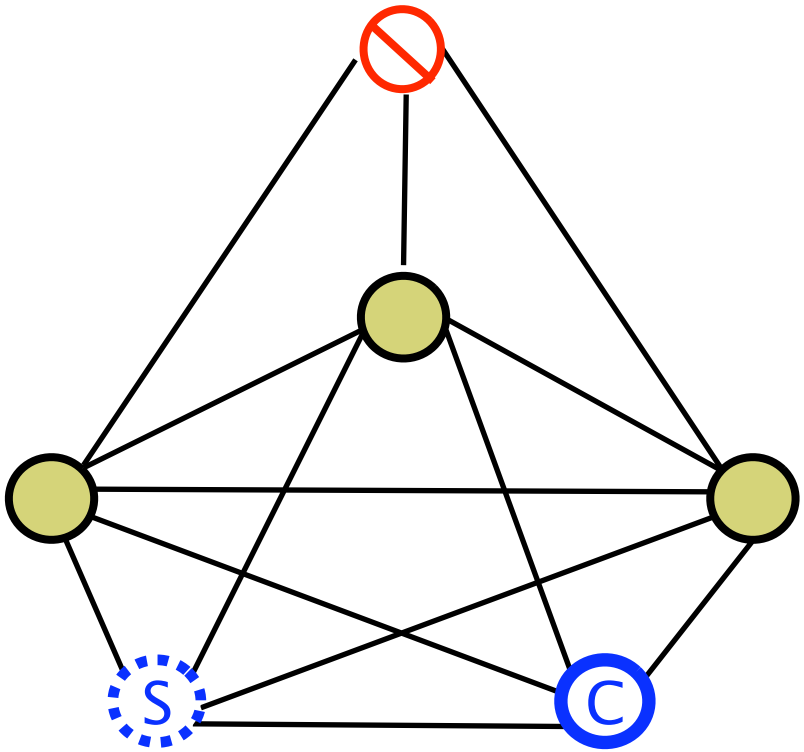
- there are no candidates anymore AND
- there are no nodes in *not* (otherwise, the recent clique is not maximal!)

The recursive procedure



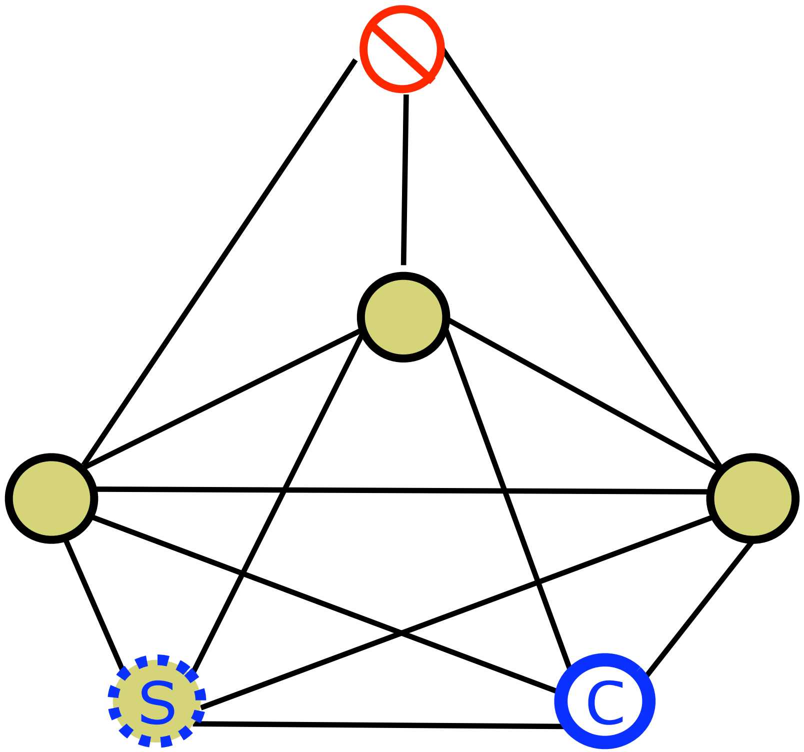
- let the *not* set consist of one node (the extension leading to the clique seen before)
- three nodes in *compsub* (known part of the clique)
- two candidates left

The recursive procedure



1. select a candidate

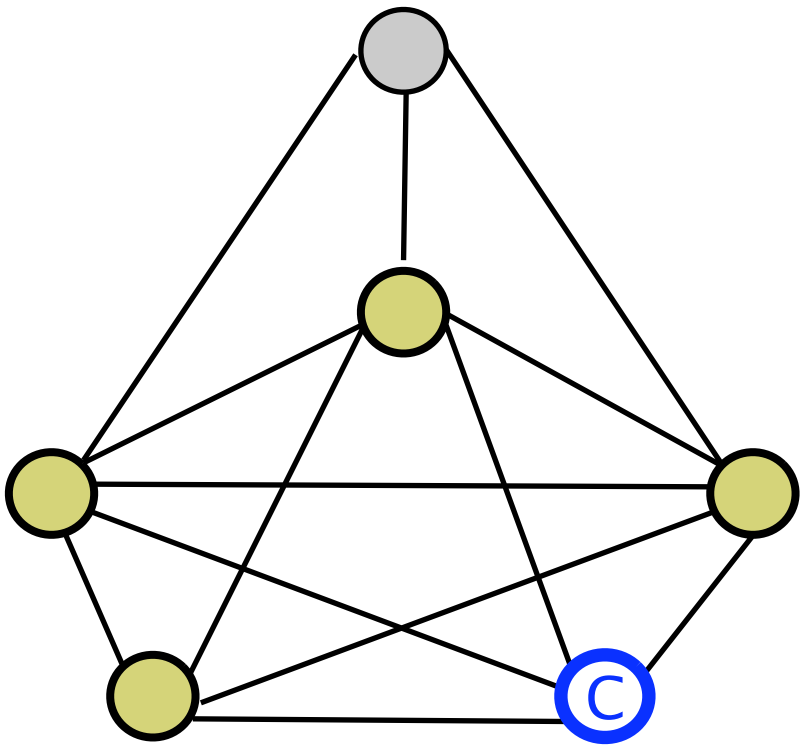
The recursive procedure



1. select a candidate

2. add it to *compsub*

The recursive procedure



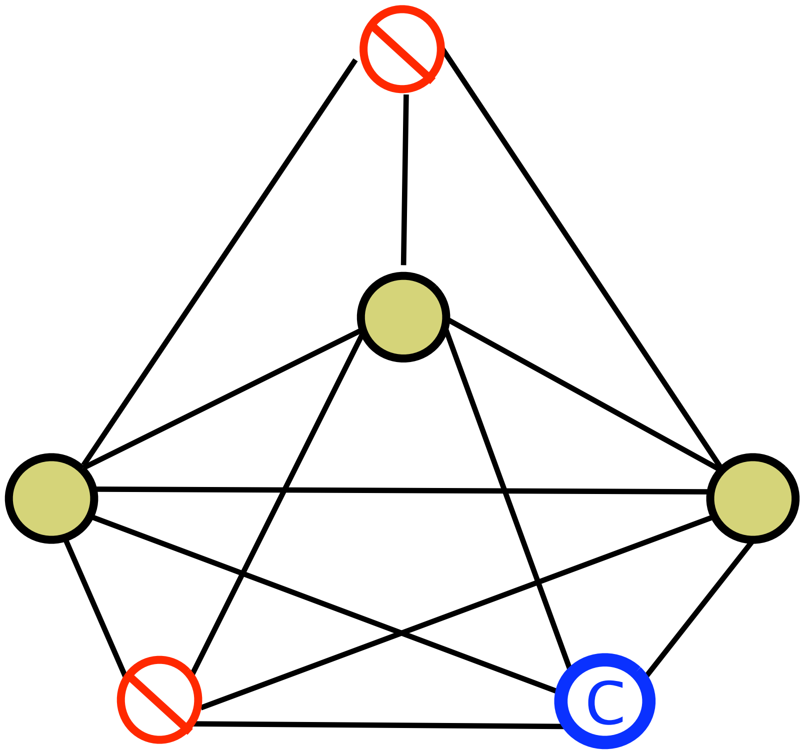
1. select a candidate

2. add it to *compsub*

3. compute new candidates and ‚not set' for the next recursion step (but store the old sets) by removing the nodes not connected to the selected candidate

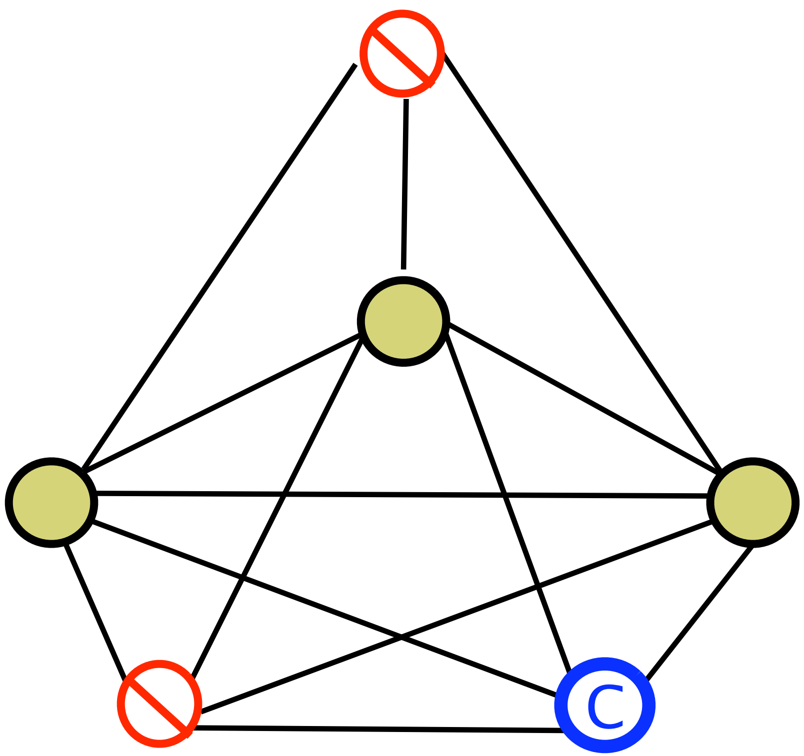
4. start at 1 with the new sets

The recursive procedure



5. back from recursion, restore the old sets and add the candidate selected before to *not*

Termination conditions



1. no candidates left or

2. there is an element in *not* which is connected to every candidate left

(if 2 holds, the addition of a candidate cannot lead to a clique which is maximal, because the node in *not* would be missing)

Optimizing candidate selection

- terminate as early as possible (minimize number of recursion steps)
- aiming at having a node in *not* connected to all candidates
- the trick:
  - nodes in *not* get a counter indicating to how many candidates they are not connected
  - pick the node in *not* with the fewest disconnections
  - in each step, pick a new candidate disconnected to this node

Outline

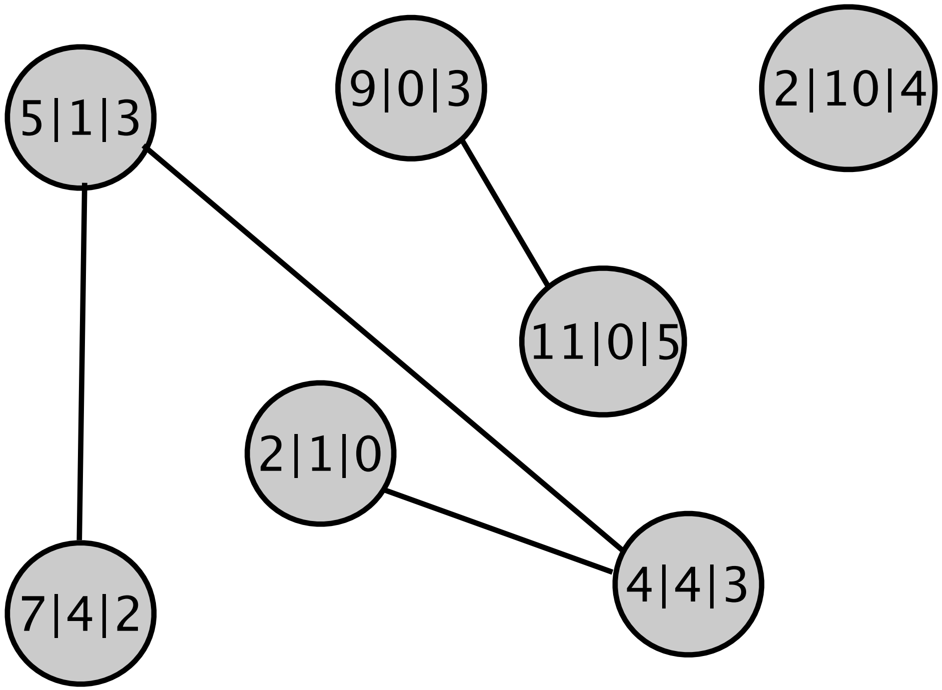
- Introduction to cliques
- The Bron-Kerbosch algorithm
- Applications
- Conclusion

Solving the maximum independent set problem

- maximum independent set of a graph: the largest set of nodes which are all connected to each other
- a ‚famous‛ application of this: compute the trunk capacity of a car
- the question here: how many blocks (of a fixed volume) fit in the trunk?

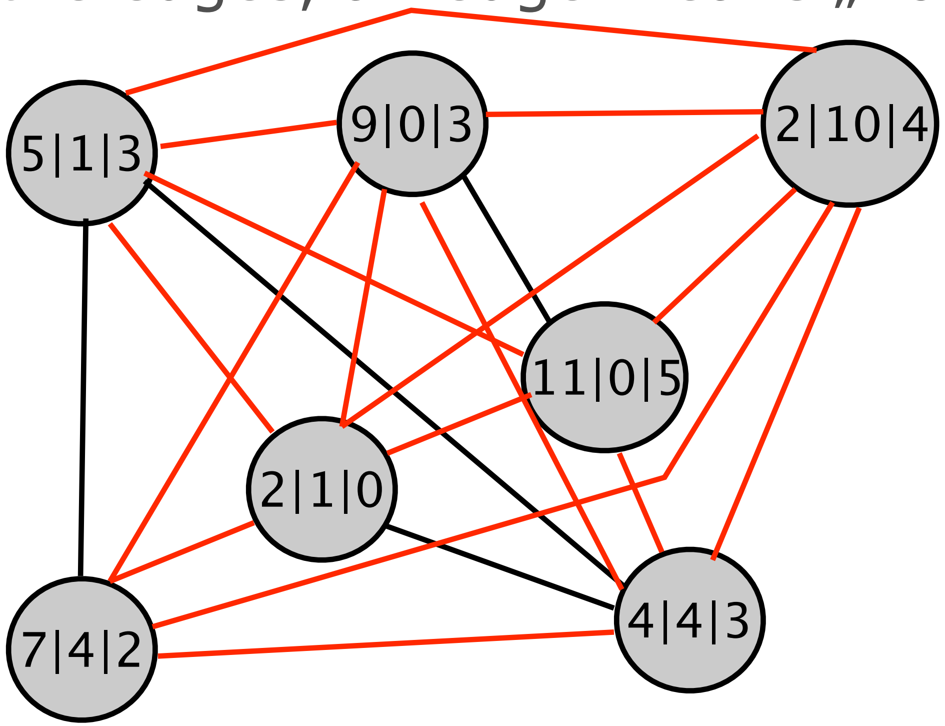
Solving the maximum independent set problem (cont.)

- nodes represent a brick and its coordinates
- an edge between two nodes means that the two bricks overlap



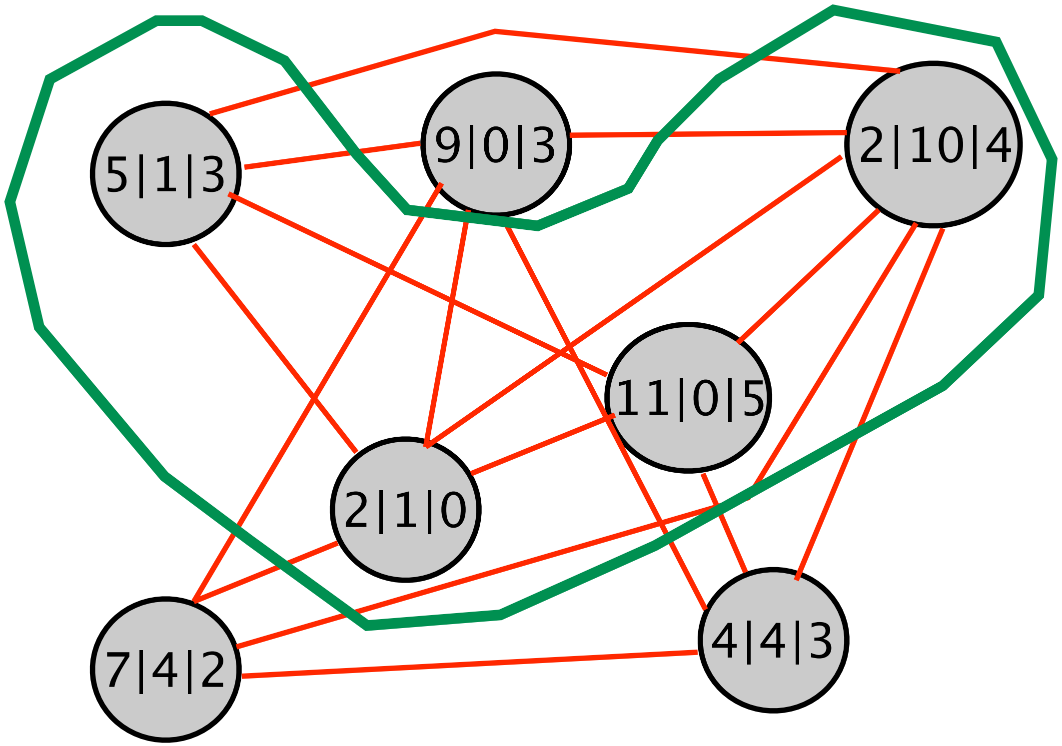
Solving the maximum independent set problem (cont.)

- the idea: bricks can be placed in the trunk at certain coordinates at the same time, if the corresponding nodes are not connected in the graph
- if we invert the edges, an edge means „not connected“

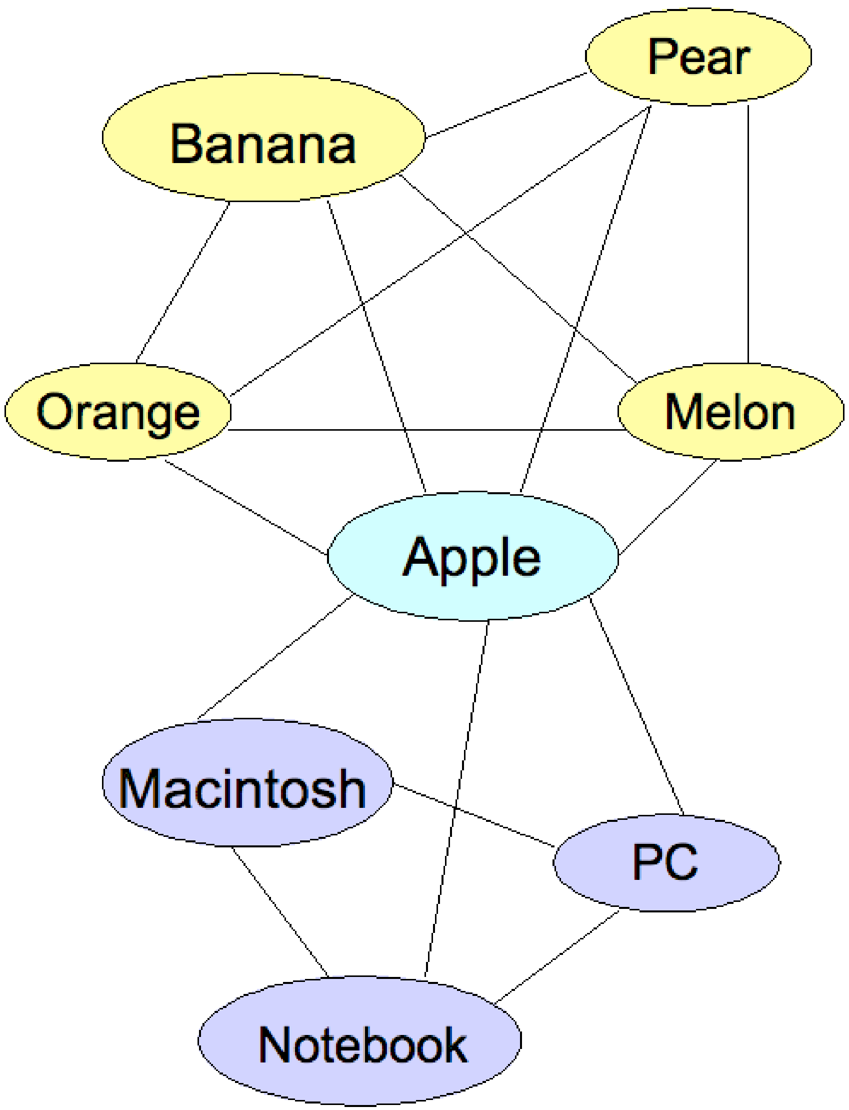


Solving the maximum independent set problem (cont.)

- now we only have to check the cliques and find the largest one(s)
- the number of nodes is the maximal number of blocks fitting



Finding different word senses?



...inspired by Widdows and Dorow (2002)

Other Applications

- several applications in bioinformatics and drug design (similarity of proteins or chemical formulas in general)
- McDonald et al. (2005) $\longrightarrow$ next talk

Conclusion

- problem: finding all cliques of a graph efficiently
- hard task (in terms of memory and runtime)
- Bron-Kerbosch algorithm is one efficient solution
- several applications - perhaps some more could be invented (operating on ontologies e.g.)

Literature

- Coen Bron and Joep Kerbosch (1973): Algorithm 457: Finding All Cliques of an Undirected Graph. Communications of the ACM Vol. 16, Issue 9. ACM Press: New York, USA.

- Dominic Widdows and Beate Dorow (2002): A graph model for unsupervised lexical acquisition. Proceedings of the 19th international conference on Computational linguistics. ACL: Morristown, USA.

- Volker Stix (2004): Finding All Maximal Cliques in Dynamic Graphs. Computational Optimization and Applications Vol. 7, Issue 2. Kluwer Academic Publishers: Norwell, USA.