Practice of Answer Set Programming

Optimization

Ira A. Fulton Schools of Engineering
Arizona State University

Objectives

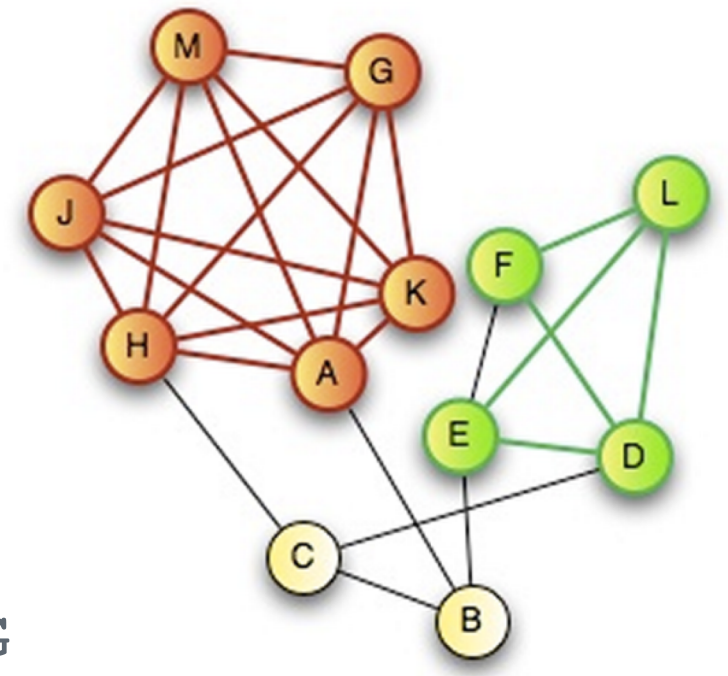
Objective

Use optimization constructs in ASP to find optimal solution

Optimization

Recall: Cliques in ASP

The stable models of the following program are in a 1–1 correspondence with cliques of cardinalities n.




```
% File 'clique'

% in/1 is a set consisting of n vertices of G
{in(X): vertex(X)} = n.

% there must be an edge between the chosen vertices
:- in(X), in(Y), X!=Y, not edge(X,Y), not edge(Y,X).
```

Maximal Clique

The directives #maximize and #minimize instruct clingo to look for the "best" stable model of the given program. The syntax is similar to aggregates

```
% choose at least n vertices
{in(X)} :- vertex(X).

% there must be an edge between the chosen vertices
:- in(X), in(Y), X!=Y, not edge(X,Y), not edge(Y,X).

#maximize{1,X: in(X)}.
#show in/1.
```

{in(1), in(2)} {(1,1) (1,2)} (2)

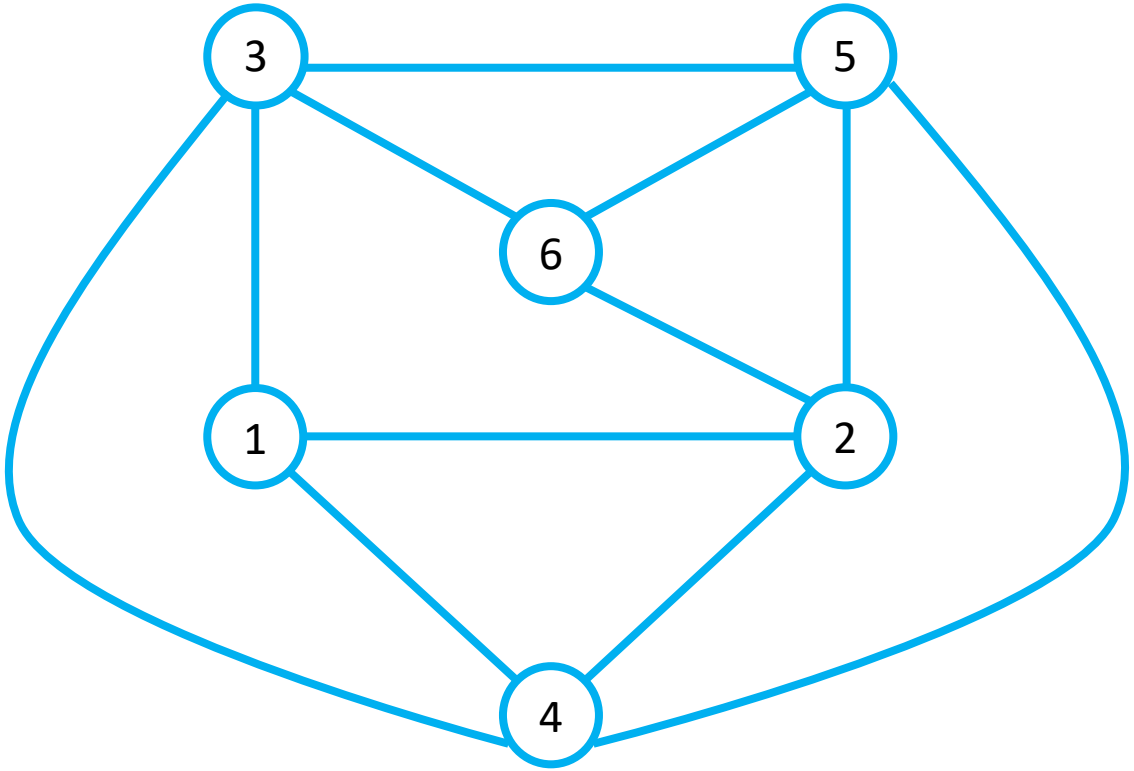
{in(1), in(3), in(5)} {(1,1), (1,3), (1,5)} (3)

Recall: Hamiltonian Cycle in ASP

```
% in(X,Y) is the set of edges included in the cycle and
% satisfies Condition 1.
{in(X,Y) : edge(X,Y)} = 1 :- vertex(X).
{in(X,Y) : edge(X,Y)} = 1 :- vertex(Y).

% Define reachability recursively
reachable(X) :- in(1,X).
reachable(Y) :- in(X,Y), reachable(X).

% Condition 2
:- not reachable(X), vertex(X).

% Display
#show in/2.
```

Recall: Graph

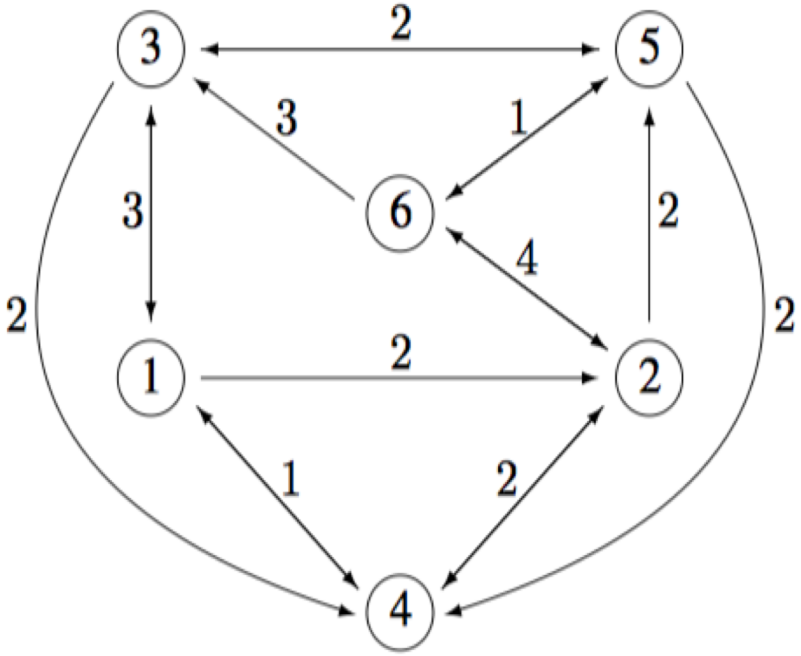
```
% Nodes
vertex(1..6).


% Edges
edge(1,(2;3;4)). edge(4,(1;2)).

edge(2,(4;5;6)). edge(5,(3;4;6)).

edge(3,(1;4;5)). edge(6,(2;3;5)).
```



Recall: Edge cost

% Edge Costs

```
cost(1,2,2). cost(2,4,2). cost(3,1,3).
cost(1,3,3). cost(2,5,2). cost(3,4,2).
cost(1,4,1). cost(2,6,4). cost(3,5,2).
cost(4,1,1). cost(5,3,2). cost(6,2,4).
cost(4,2,2). cost(5,4,2). cost(6,3,3).
cost(5,6,1). cost(6,5,1).
```



% Optimize

```
#minimize { C,X,Y : cycle(X,Y), cost(X,Y,C) }.
```

Running clingo

% clingo graph1.lp ham-edge-cost.lp ham.lp ham-optimize.lp 0

```
Solving...
  Answer: 1
  cycle(1,4) cycle(4,2) cycle(3,1) cycle(2,6) cycle(6,5) cycle(5,3)
  Optimization: 13
  Answer: 2
  cycle(1,4) cycle(4,2) cycle(3,1) cycle(2,5) cycle(6,3) cycle(5,6)
  Optimization: 12
  Answer: 3
  cycle(1,2) cycle(4,1) cycle(3,4) cycle(2,5) cycle(6,3) cycle(5,6)
  Optimization: 11
  OPTIMUM FOUND
```

Assigning Referees

Assigning Papers to Referees: Context

Research papers submitted to a technical conference are usually reviewed by the conference program committee. Every submission is read and discussed by a group of committee members chosen by the chair of the committee, and this group decides if the paper can be accepted for presentation.

Assigning Papers to Referees: Criteria

To help the chair find a good match between submissions and referees, every committee member submits a bid that classifies all submissions into three categories: “yes” (I want to review this submission), “maybe” (I don’t mind reviewing it), and “no”: (don’t assign it to me). The chair tries to assign each submission for review to a specific number k of committee members so that

- the workloads are approximately equal;
- no committee member is asked to review a submission that he placed in the “no” group;
- the total number of cases when a submission is assigned to a reviewer who placed it in the “yes” group is as large as possible.

Assign Papers to Referees: Process

Input

- k: number of referees per submission
- referee/1: set of committee members
- submission/1: set of submissions;
- bid/3: set of triples (R,S,B) such that B is the R's bid for submission S.

% every submission is assigned to k referees who don't mind reviewing it.

```
{review(R,S):referee(R), not bid(R,S,no)} = k
                                    :-   submission(S).
```

Assign Papers to Referees: Process, cont'd

% workload(R,N) iff N is the number of submissions assigned to referee R.

```
workload(R,N) :- referee(R), N = #count{S : review(R,S)}.
```

% the difference between the workloads of committee members is at most 1.

```
:- workload(R1,N1), workload(R2,N2), |N1 -N2| > 1.
```

% the number of "yes" cases is maximal.

```
#maximize {1,R,S : bid(R,S,yes), review(R,S)}.
```

Wrap-Up