Use cases

Bolisetty Lokesh

Control Flow

1

STEP 1

Decice the problem

Control Flow

1

STEP 1

Decice the problem

2

STEP 2

List the variables and constraints in a .yaml file

```
3 lights in the kitchen
3 lights in the living room
3 TVs
2 dining hall lights

model in kitchen : 0 if 0.6 * l_k1 + 0.6 * l_k2 + 0.6 * l_k3 + 0.3 * l_lv2
+ 0.3 * l_tv3 == mv_kitchen else 1000

Rules:
10 * (abs(mv_livingroom - 5) + abs(mv_kitchen - 4))
```

Control Flow

1

STEP 1

Decice the problem

2

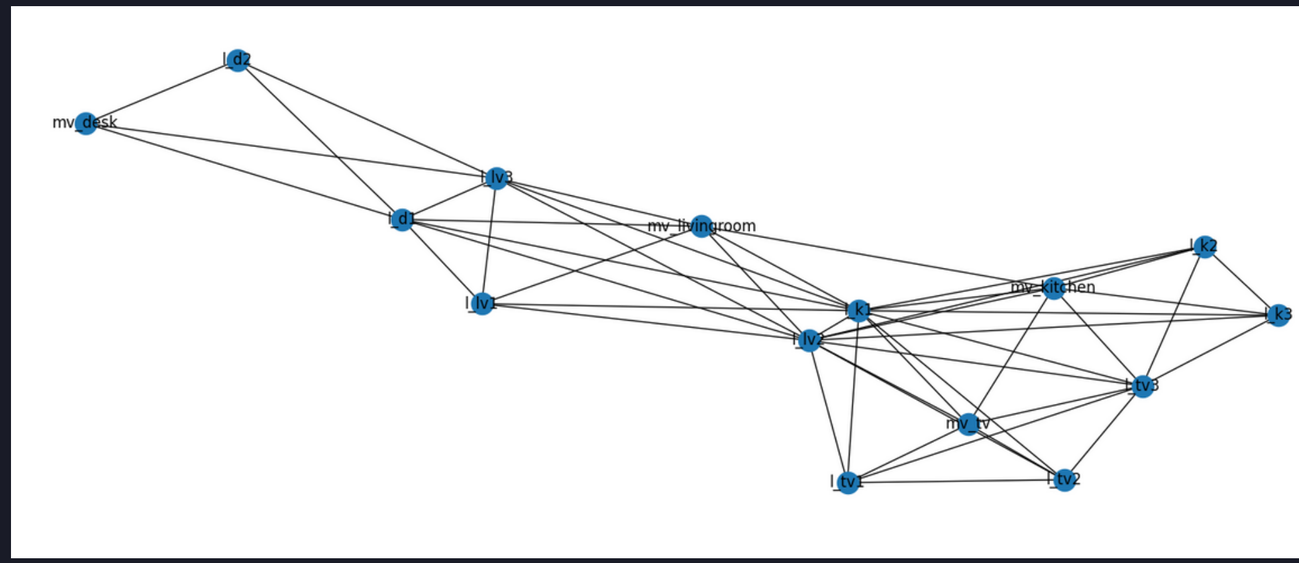
STEP 2

List the variables and constraints in a .yaml file

3

STEP 3

Generate the graph



Control Flow

1

STEP 1

Decice the problem

2

STEP 2

List the variables and constraints in a .yaml file

3

STEP 3

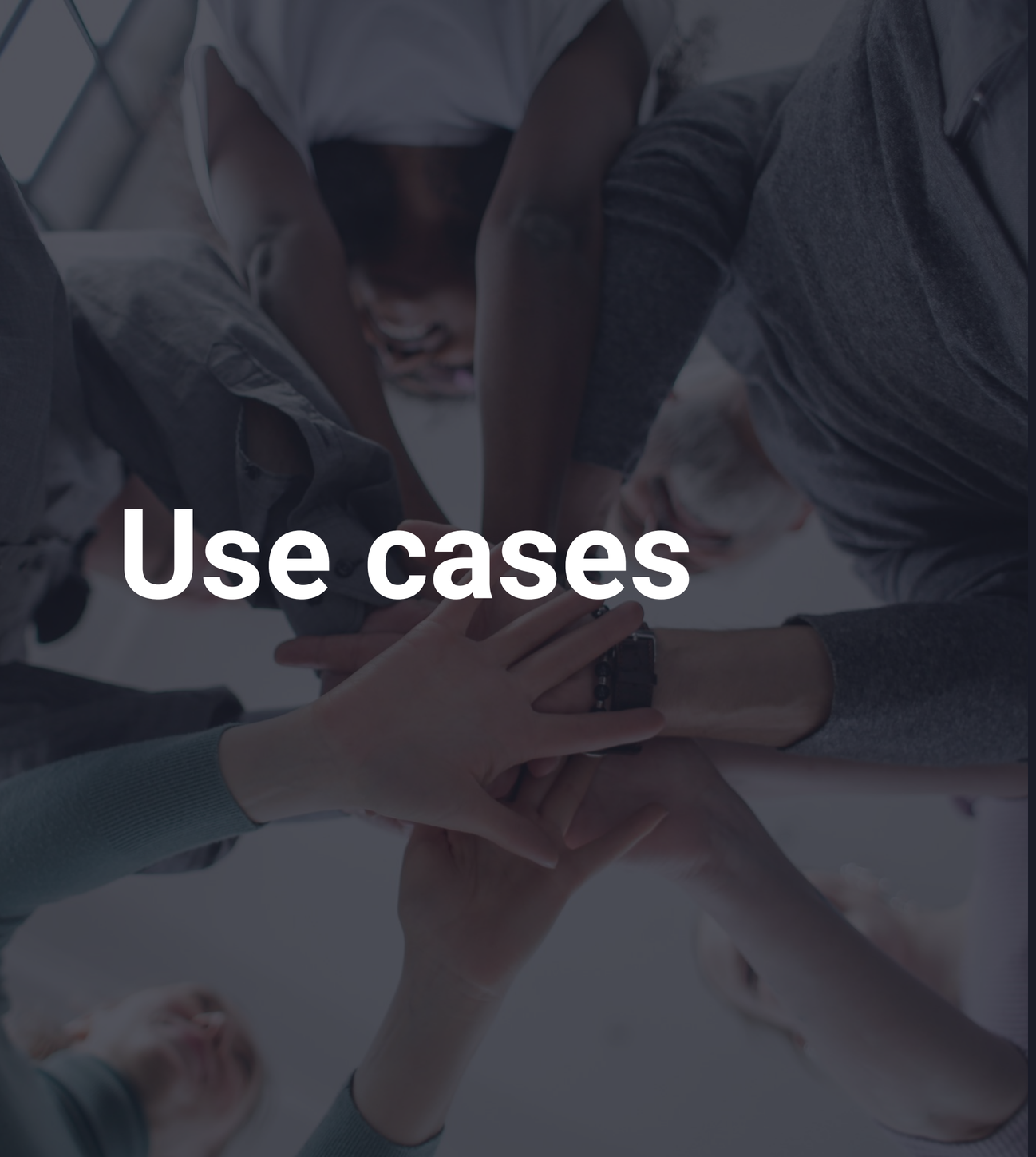
Generate the graph

4

STEP 4

Chose suitable algorithm and find the best assignment


```
"assignment": {
  "l_d1": 0,
  "l_d2": 0,
  "l_e1": 0,
  "l_e2": 0,
  "l_k1": 0,
  "l_k2": 0,
  "l_k3": 0,
  "l_lv1": 0,
  "l_lv2": 0,
  "l_lv3": 0,
  "l_tv1": 0,
  "l_tv2": 0,
  "l_tv3": 0,
  "mv_desk": 4,
  "mv_entry": 4,
  "mv_kitchen": 4,
  "mv_livingroom": 0,
  "mv_stairs": 0,
  "mv_tv": 0
},
```



Use cases

- SECP
- MEETING SCHEDULER
- GRAPH COLOURING
- ISING AND OTHERS

SECP

NEXT

Our Mission

Smart Environment Configuration problem

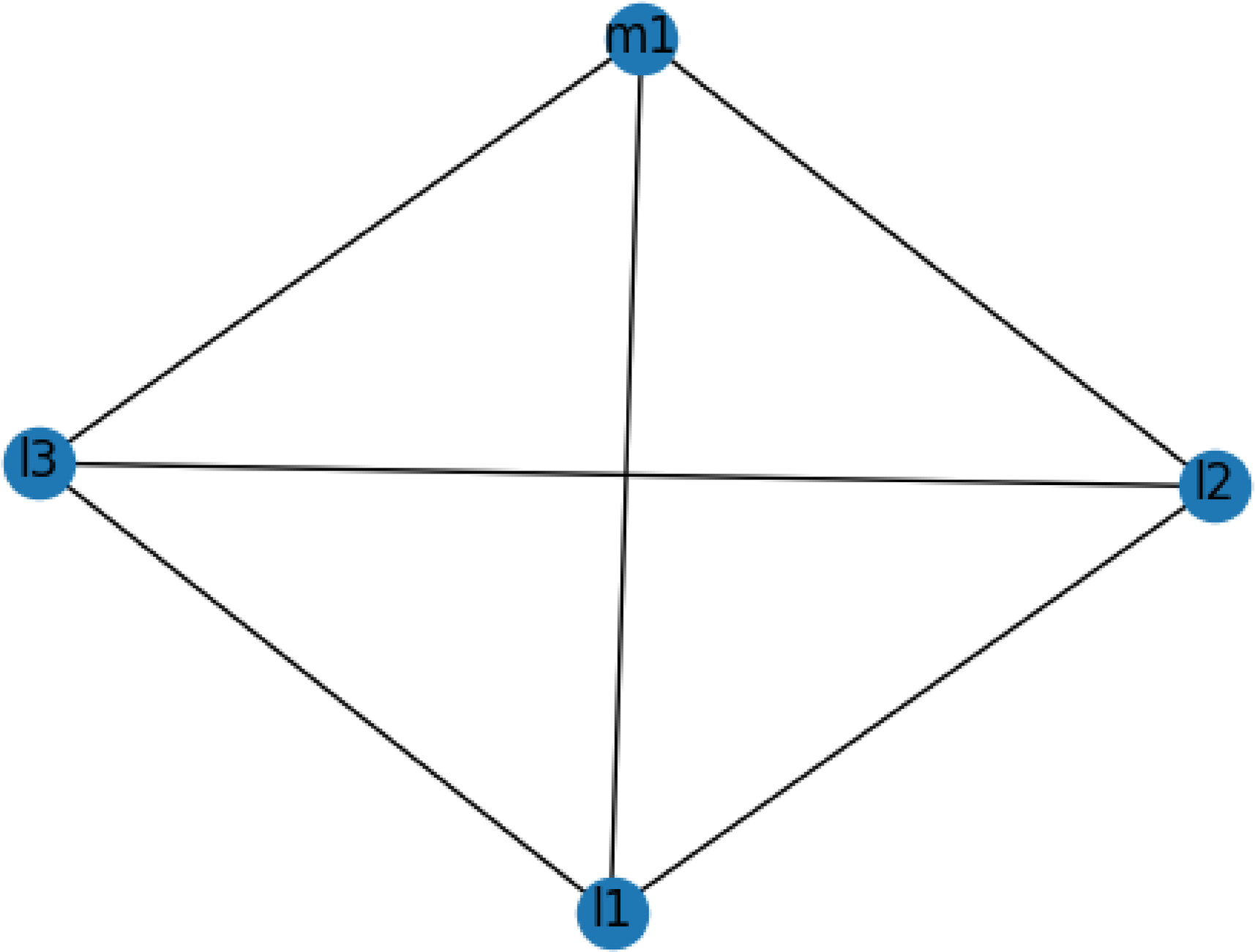
Required Attributes

1. All appliances like lights fans in the environment
2. Their attributes like speed of fan, luminosity
3. Physical systems that depend on the appliances
4. Rules that are to be followed

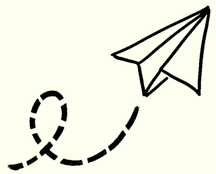
Example

```yaml
domains:
  luminosity:
    values: [0, 1, 2, 3, 4]
    type: 'luminosity'

variables:
# l1, l2 and l3 represent the 3 light bulb. They each have a different
# efficiency (from their cost function), l3 being the most efficient and l1
# the less efficient
  l1:
    domain: luminosity
    cost_function: 0.7 * l1
  l2:
    domain: luminosity
    cost_function: 0.5 * l2
  l3:
    domain: luminosity
    cost_function: 0.2 * l3

# m1 is the variable associated to the physical model that depends on l1, l2
 # and l2
  m1:
    domain: luminosity


constraints:
# m1_c is the contraint that bind the light bulb to the physical model
# variable m1
  m1_c:
    type: intention
    function: 0 if m1 == round(0.7 * l1 + 0.5 * l2 + 0.3 * l3) else 1000
# r1 is the constraint model the user rule: the target is to have a
# luminosity of 3 for the physical model and 2 for l2
  r1:
    type: intention
    function: 10 * (abs(m1 - 3) + abs(l2 -3))
```

Generate the graph

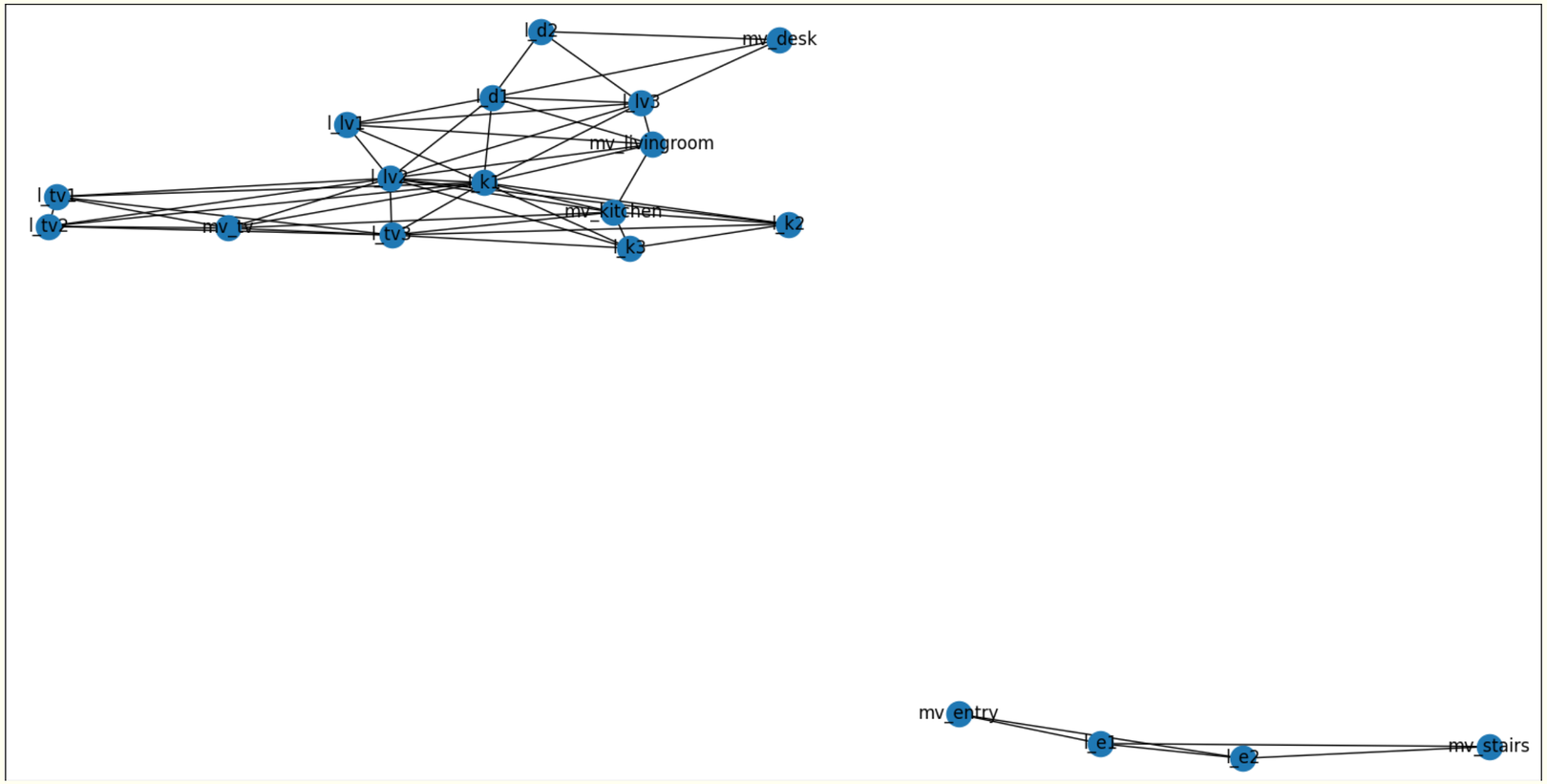


Run to find the suitable assignment




```
"assignment": {
    "l1": 0,
    "l2": 0,
    "l3": 0,
    "m1": 3
},
"cost": 1030.0,
"cycle": 0,
"msg_count": 28,
"msg_size": 10,
"status": "TIMEOUT",
"time": 3.003302949000499,
"violation": 0
```

A more complicated example




```
},
"assignment": {
  "l_d1": 0,
  "l_d2": 0,
  "l_e1": 0,
  "l_e2": 0,
  "l_k1": 0,
  "l_k2": 0,
  "l_k3": 0,
  "l_lv1": 0,
  "l_lv2": 0,
  "l_lv3": 0,
  "l_tv1": 0,
  "l_tv2": 0,
  "l_tv3": 0,
  "mv_desk": 4,
  "mv_entry": 4,
  "mv_kitchen": 4,
  "mv_livingroom": 0,
  "mv_stairs": 0,
  "mv_tv": 0
},
"cost": 3111.2589174460463,
"cycle": 0,
"msg_count": 145,
"msg_size": 60,
"status": "TIMEOUT",
"time": 3.008901335997507,
"violation": 0
```

Meeting Scheduler

NEXT

Our Mission

Schedule meetings with least cost with given resources

Required Attributes

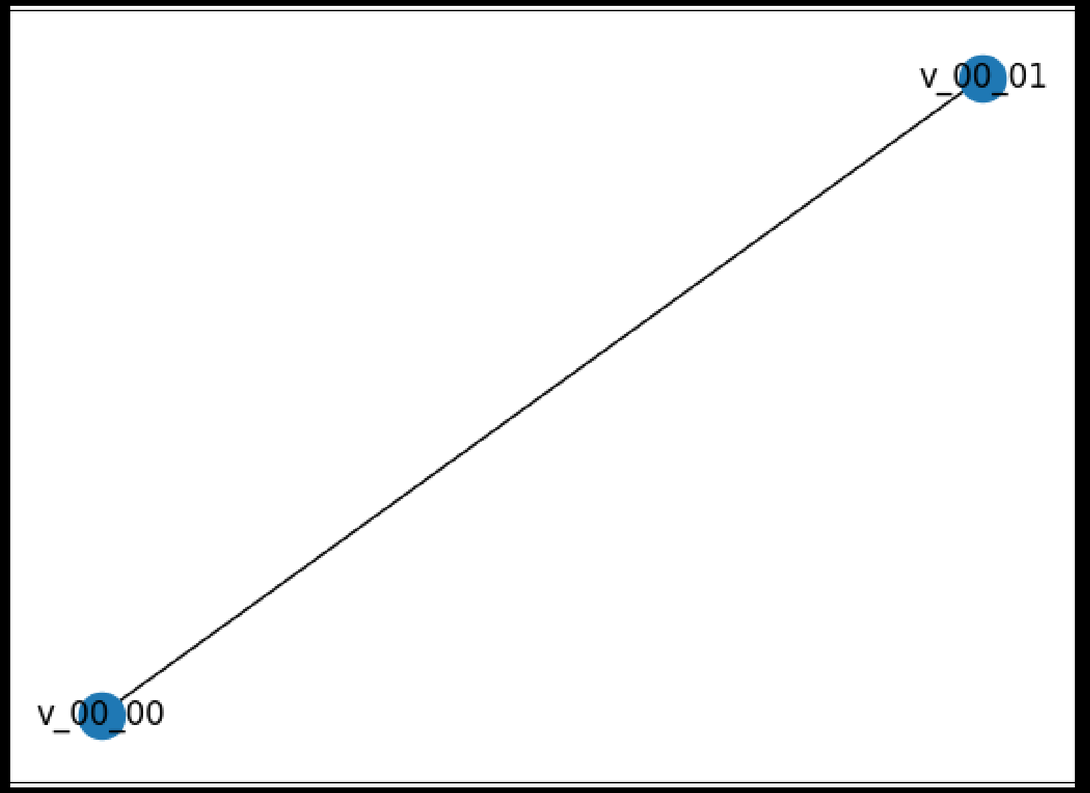
1. Number of time slots available
2. Number of resources(like classrooms) available
3. Number of meetings to be scheduled
4. Max number of resources for one meeting
5. Max number of slots for one meeting

pydcop --output meetingsExample.yaml generate meetings --slots_count 6 --events_count 2 --resources_count 1 --max_resources_event 1

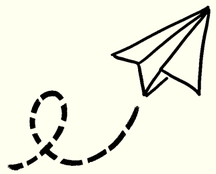
Example

```yaml
variables:
  v_00_00:
    domain: d_v_00_00
  v_00_01:
    domain: d_v_00_01

constraints:
  ci_v_00_00_v_00_01:
    type: extensional
    values:
      -60.0: 1 1 | 2 2 | 3 3 | 4 4 | 5 5 | 6 6
      -8.0: 2 1 | 1 2
      -7.0: 1 0 | 2 0
      -4.0: 6 1 | 6 2 | 1 6 | 2 6
      -3.0: 6 0 | 4 1 | 5 1 | 4 2 | 5 2 | 1 4 | 2 4 | 1 5 | 2 5
      -2.0: 4 0 | 5 0
      -1.0: 0 1 | 3 1 | 0 2 | 3 2 | 1 3 | 2 3
      0.0: 0 0 | 3 0
      1.0: 6 4 | 6 5 | 4 6 | 5 6
      2.0: 5 4 | 4 5
      3.0: 6 3 | 0 6 | 3 6
      4.0: 4 3 | 5 3 | 0 4 | 3 4 | 0 5 | 3 5
      6.0: 0 3
    variables:
    - v_00_00
    - v_00_01

agents:
  a_0: {}
hosting_costs:
  a_0:
    computations:
      v_00_00: 0
      v_00_01: 0
    default: 0
routes:
  default: 1
```

Generate the graph



Run to find the suitable assignment




```
{
  "agt_metrics": {
    "a_0": {
      "activity_ratio": 1.5999383157225318,
      "count_ext_msg": {
        "_discovery_a_0": 7
      },
      "cycles": {
        "v_00_00": 0,
        "v_00_01": 0
      },
      "size_ext_msg": {
        "_discovery_a_0": 0
      }
    }
  },
  "assignment": {
    "v_00_00": 0,
    "v_00_01": 3
  },
  "cost": 6.0,
  "cycle": 0,
  "msg_count": 7,
  "msg_size": 0,
  "status": "FINISHED",
  "time": 0.0054026720026740 8,
  "violation": 0
}
```

Other examples

1. Ising
2. Small World graph
3. Graph colouring

Other examples

1. Register allocation - compiler
2. Frequency allocation to cell phone towers


Thank you!