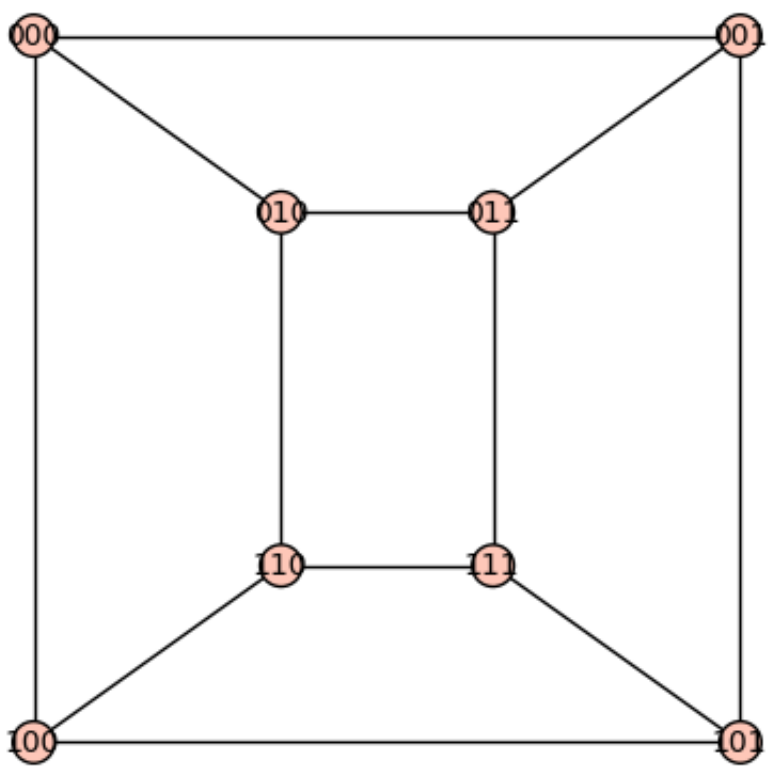
MATH 482: TSP

Here, we build a graph and weights on the edges.

```
G = graphs.CubeGraph(3);
plt = {};
plt['000'] = (-10, 10);
plt['100'] = (-10, -10);
plt['010'] = (-3, 5);
plt['110'] = (-3, -5);
plt['101'] = (10, -10);
plt['001'] = (10, 10);
plt['111'] = (3, -5);
plt['011'] = (3, 5);
G.set_pos(plt);
G.show();
print G.edges();
```

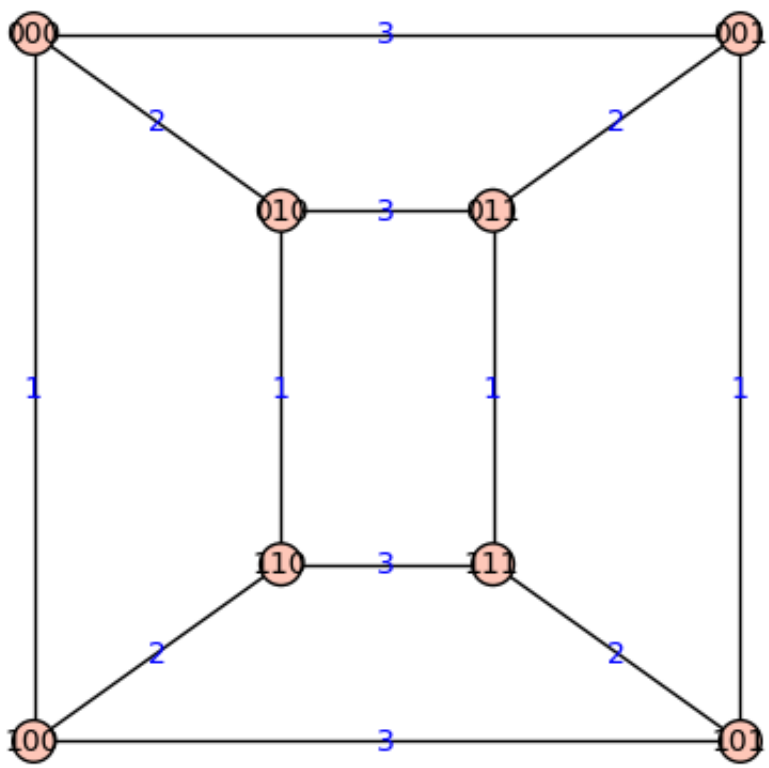



```
[('000', '001', None), ('000', '010', None), ('000', '100', None
('001', '011', None), ('001', '101', None), ('010', '011', None
('010', '110', None), ('011', '111', None), ('100', '101', None
('100', '110', None), ('101', '111', None), ('110', '111', None
```

We now assign weights to the edges.

```
def StringDiff(u, v):
    l = min(len(u), len(v));
    count = 0;
    for i in range(l):
        if u[i] != v[i]:
            count = i;
    return count+1;

w = {};
for e in G.edges():
    u,v,l = e;
    w[(u,v)] = StringDiff(u,v);
    G.set_edge_label(u, v, w[(u,v)]);
print G.edges();
G.show(edge_labels=True);
```

```
[('000', '001', 3), ('000', '010', 2), ('000', '100', 1), ('001
'011', 2), ('001', '101', 1), ('010', '011', 3), ('010', '110',
('011', '111', 1), ('100', '101', 3), ('100', '110', 2), ('101'
'111', 2), ('110', '111', 3)]
```



We now build our ILP with no "cut constraints"

```
p = MixedIntegerLinearProgram(maximization=False);

x = p.new_variable();
edge_labels = {};
```

```
ecount = 0;

for e in G.edges():
    edge_labels[(e[0],e[1])] = ecount;
    p.set_integer(x[ecount]);
    p.set_min(x[ecount],0);
    p.set_max(x[ecount],1);
    ecount += 1;

    edge_labels[(e[1],e[0])] = ecount;
    p.set_integer(x[ecount]);
    p.set_min(x[ecount],0);
    p.set_max(x[ecount],1);
    ecount += 1;

for u in G.vertices():
    p.add_constraint( sum( x[edge_labels[(u,v)]] for v in
G.neighbors(u) ) == 1 );
    p.add_constraint( sum( x[edge_labels[(v,u)]] for v in
G.neighbors(u) ) == 1 );

p.set_objective( sum( w[(e[0],e[1])] *
x[edge_labels[(e[0],e[1])]] for e in G.edges() ) + sum(
w[(e[0],e[1])] * x[edge_labels[(e[1],e[0])]] for e in G.edges() )
);
```

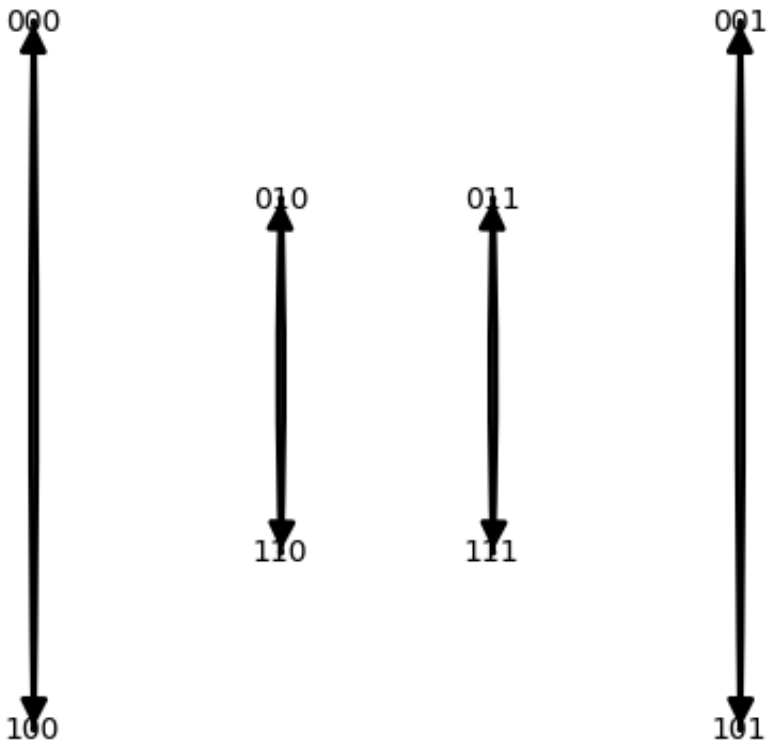
```
p.show()
```

```
Minimization:
  3.0 x_0 +3.0 x_1 +2.0 x_2 +2.0 x_3 +x_4 +x_5 +2.0 x_6 +2.0 x_
+x_8 +x_9 +3.0 x_10 +3.0 x_11 +x_12 +x_13 +x_14 +x_15 +3.0 x_16
x_17 +2.0 x_18 +2.0 x_19 +2.0 x_20 +2.0 x_21 +3.0 x_22 +3.0 x_2
Constraints:
  1.0 <= x_0 +x_2 +x_4 <= 1.0
  1.0 <= x_1 +x_3 +x_5 <= 1.0
  1.0 <= x_1 +x_6 +x_8 <= 1.0
  1.0 <= x_0 +x_7 +x_9 <= 1.0
  1.0 <= x_3 +x_10 +x_12 <= 1.0
  1.0 <= x_2 +x_11 +x_13 <= 1.0
  1.0 <= x_7 +x_11 +x_14 <= 1.0
  1.0 <= x_6 +x_10 +x_15 <= 1.0
  1.0 <= x_5 +x_16 +x_18 <= 1.0
  1.0 <= x_4 +x_17 +x_19 <= 1.0
  1.0 <= x_9 +x_17 +x_20 <= 1.0
  1.0 <= x_8 +x_16 +x_21 <= 1.0
  1.0 <= x_13 +x_19 +x_22 <= 1.0
  1.0 <= x_12 +x_18 +x_23 <= 1.0
  1.0 <= x_15 +x_21 +x_23 <= 1.0
```

```
    1.0 <= x_14 +x_20 +x_22 <= 1.0
  Variables:
    x_0 is a boolean variable (min=0.0, max=1.0)
    x_1 is a boolean variable (min=0.0, max=1.0)
    x_2 is a boolean variable (min=0.0, max=1.0)
    x_3 is a boolean variable (min=0.0, max=1.0)
    x_4 is a boolean variable (min=0.0, max=1.0)
    x_5 is a boolean variable (min=0.0, max=1.0)
    x_6 is a boolean variable (min=0.0, max=1.0)
    x_7 is a boolean variable (min=0.0, max=1.0)
    x_8 is a boolean variable (min=0.0, max=1.0)
    x_9 is a boolean variable (min=0.0, max=1.0)
    x_10 is a boolean variable (min=0.0, max=1.0)
    x_11 is a boolean variable (min=0.0, max=1.0)
    x_12 is a boolean variable (min=0.0, max=1.0)
    x_13 is a boolean variable (min=0.0, max=1.0)
    x_14 is a boolean variable (min=0.0, max=1.0)
    x_15 is a boolean variable (min=0.0, max=1.0)
    x_16 is a boolean variable (min=0.0, max=1.0)
    x_17 is a boolean variable (min=0.0, max=1.0)
    x_18 is a boolean variable (min=0.0, max=1.0)
    x_19 is a boolean variable (min=0.0, max=1.0)
    x_20 is a boolean variable (min=0.0, max=1.0)
    x_21 is a boolean variable (min=0.0, max=1.0)
    x_22 is a boolean variable (min=0.0, max=1.0)
    x_23 is a boolean variable (min=0.0, max=1.0)
```

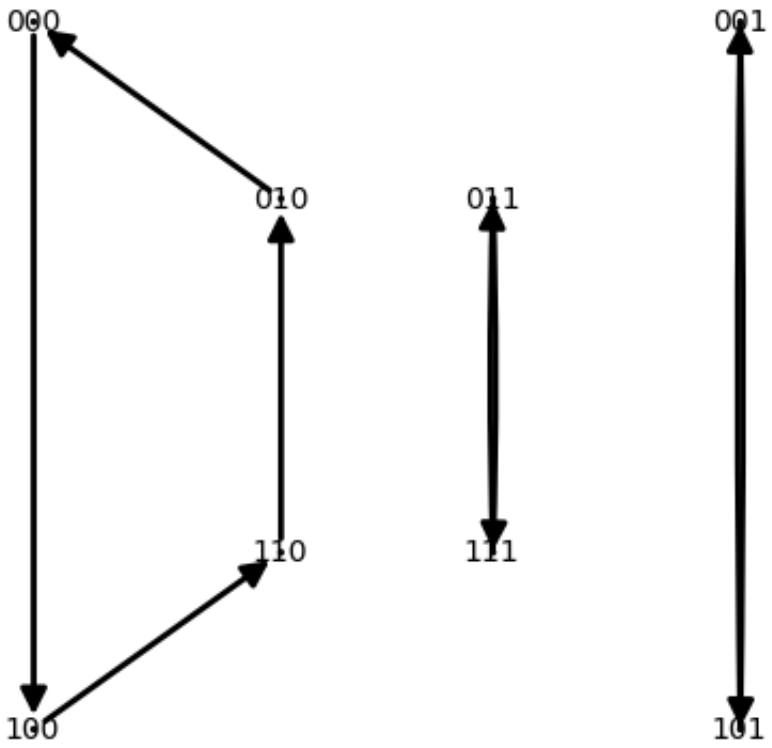
We solve and look at the tour.

```
p.solve();
vals = p.get_values(x);
H = DiGraph();
H.set_pos(plt);

for u,v,l in G.edges():
    if vals[ edge_labels[ (u,v) ] ] > 0:
        H.add_edge(u,v);
    if vals[ edge_labels[ (v,u) ] ] > 0:
        H.add_edge(v,u);
H.show();
```

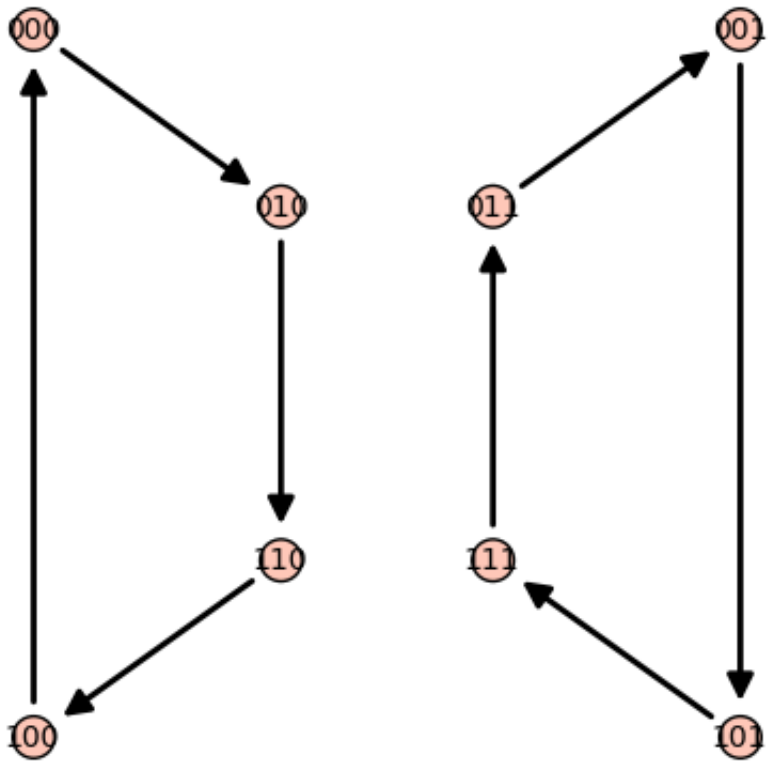


We must add a cut contraint.

```
S = [  '000', '100'];
Sb = [  '010', '110', '111', '011', '101', '001' ];
elist = [];
for u in S:
    for v in Sb:
        if G.has_edge(u,v):
            elist.append( (u,v) );

p.add_constraint( sum( x[edge_labels[e]] for e in elist) >= 1 );
p.solve();
vals = p.get_values(x);
H = DiGraph();
H.set_pos(plt);

for u,v,l in G.edges():
    if vals[ edge_labels[ (u,v) ] ] > 0:
        H.add_edge(u,v);
    if vals[ edge_labels[ (v,u) ] ] > 0:
        H.add_edge(v,u);
H.show();
```

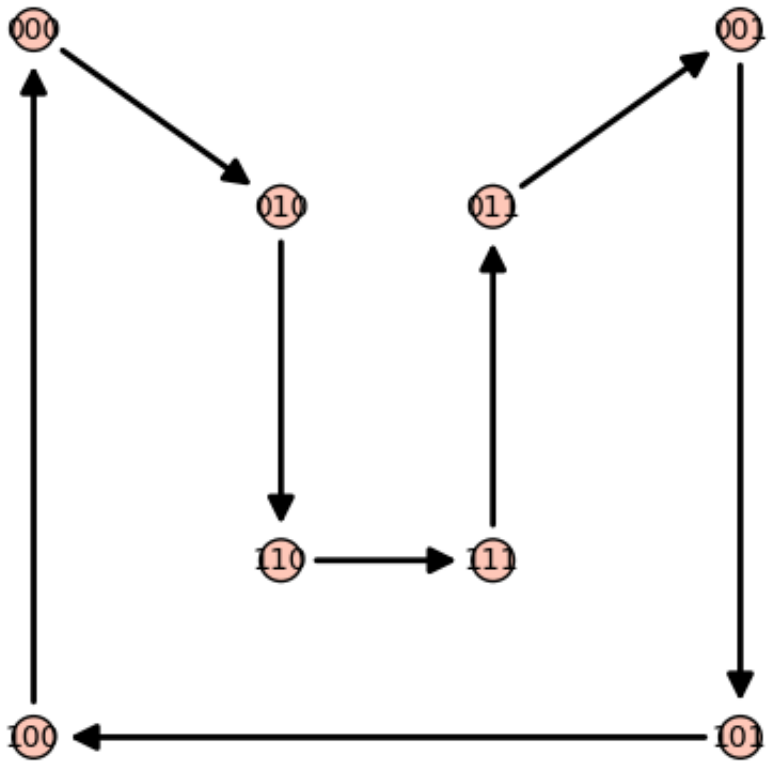



```
S = [   '111', '011'];
Sb = [  '010', '110', '101', '001','000', '100' ];
elist = [];
for u in S:
    for v in Sb:
        if G.has_edge(u,v):
            elist.append( (u,v) );

p.add_constraint( sum( x[edge_labels[e]] for e in elist) >= 1 );
p.solve();
vals = p.get_values(x);
H = DiGraph();
H.set_pos(plt);

for u,v,l in G.edges():
    if vals[ edge_labels[ (u,v) ] ] > 0:
        H.add_edge(u,v);
    if vals[ edge_labels[ (v,u) ] ] > 0:
        H.add_edge(v,u);
H.show();
```



We solve and look at the tour.

```
S = [  '000', '100', '010', '110' ];
Sb = [ '111', '011', '101', '001' ];
elist = [];
for u in S:
    for v in Sb:
        if G.has_edge(u,v):
            elist.append( (u,v) );

p.add_constraint( sum( x[edge_labels[e]] for e in elist) >= 1 );

p.solve();
vals = p.get_values(x);
H = DiGraph();
H.set_pos(plt);

for u,v,l in G.edges():
    if vals[ edge_labels[ (u,v) ] ] > 0:
        H.add_edge(u,v);
    if vals[ edge_labels[ (v,u) ] ] > 0:
        H.add_edge(v,u);
H.show();
```



This is a solution!

We now use the standalone method to solve in one shot!

```
p = MixedIntegerLinearProgram(maximization=False);

n = G.num_verts();
x = p.new_variable();
y = p.new_variable();
edge_labels = {};

ecount = 0;

for e in G.edges():
    edge_labels[(e[0],e[1])] = ecount;
    p.set_integer(x[ecount]);
    p.set_min(x[ecount],0);
    p.set_max(x[ecount],1);
    ecount += 1;

    edge_labels[(e[1],e[0])] = ecount;
    p.set_integer(x[ecount]);
    p.set_min(x[ecount],0);
    p.set_max(x[ecount],1);
    ecount += 1;
```

```
    p.add_constraint( x[ecount - 2] + x[ecount - 1] <= 1 ); # at
most one of each

for u in G.vertices():
    p.set_min(y[u], None);
    p.set_max(y[u], None);
    p.add_constraint( sum( x[edge_labels[(u,v)]] for v in
G.neighbors(u) ) == 1 );
    p.add_constraint( sum( x[edge_labels[(v,u)]] for v in
G.neighbors(u) ) == 1 );
    if u != '000':
        for v in G.neighbors(u):
            if v != '000':
                p.add_constraint( y[u] - y[v] + (n-
1)*x[edge_labels[(u,v)]] <= n - 2 );
                p.add_constraint( y[v] - y[u] + (n-
1)*x[edge_labels[(v,u)]] <= n - 2 );

p.set_objective( sum( w[(e[0],e[1])] *
x[edge_labels[(e[0],e[1])]] for e in G.edges() ) + sum(
w[(e[0],e[1])] * x[edge_labels[(e[1],e[0])]] for e in G.edges() )
);
```

```
p.show()
```

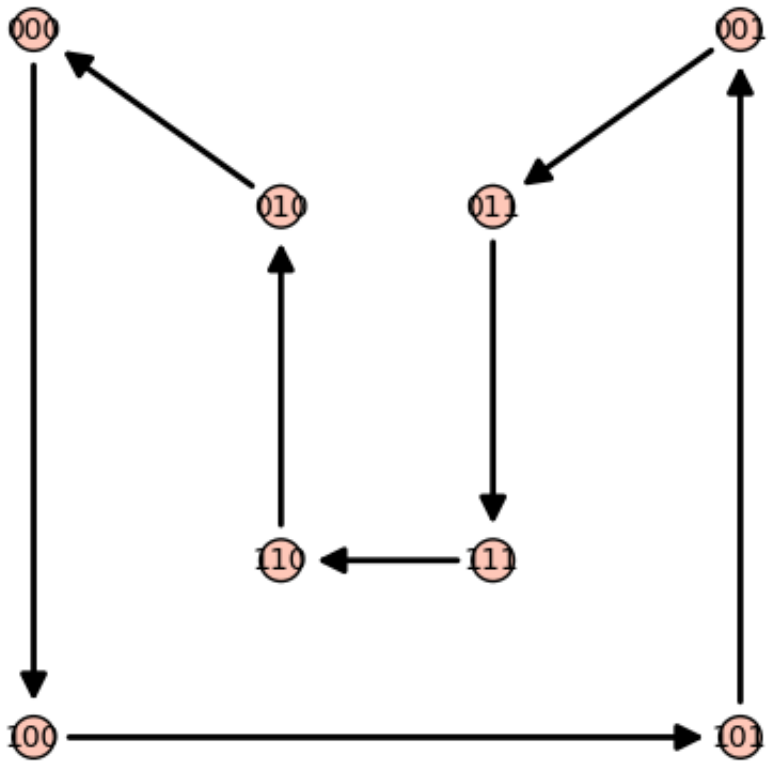
```
Minimization:
  3.0 x_0 +3.0 x_1 +2.0 x_2 +2.0 x_3 +x_4 +x_5 +2.0 x_6 +2.0 x_
+x_8 +x_9 +3.0 x_10 +3.0 x_11 +x_12 +x_13 +x_14 +x_15 +3.0 x_16
x_17 +2.0 x_18 +2.0 x_19 +2.0 x_20 +2.0 x_21 +3.0 x_22 +3.0 x_2
Constraints:
  x_0 +x_1 <= 1.0
  x_2 +x_3 <= 1.0
  x_4 +x_5 <= 1.0
  x_6 +x_7 <= 1.0
  x_8 +x_9 <= 1.0
  x_10 +x_11 <= 1.0
  x_12 +x_13 <= 1.0
  x_14 +x_15 <= 1.0
  x_16 +x_17 <= 1.0
  x_18 +x_19 <= 1.0
  x_20 +x_21 <= 1.0
  x_22 +x_23 <= 1.0
  1.0 <= x_0 +x_2 +x_4 <= 1.0
  1.0 <= x_1 +x_3 +x_5 <= 1.0
  1.0 <= x_1 +x_6 +x_8 <= 1.0
  1.0 <= x_0 +x_7 +x_9 <= 1.0
  7.0 x_6 +x_25 -x_26 <= 6.0
  7.0 x_7 -x_25 +x_26 <= 6.0
```

```
  7.0 x_8 +x_25 -x_27 <= 6.0
  7.0 x_9 -x_25 +x_27 <= 6.0
  1.0 <= x_3 +x_10 +x_12 <= 1.0
  1.0 <= x_2 +x_11 +x_13 <= 1.0
  7.0 x_10 -x_26 +x_28 <= 6.0
  7.0 x_11 +x_26 -x_28 <= 6.0
  7.0 x_12 +x_28 -x_29 <= 6.0
  7.0 x_13 -x_28 +x_29 <= 6.0
  1.0 <= x_7 +x_11 +x_14 <= 1.0
  1.0 <= x_6 +x_10 +x_15 <= 1.0
  7.0 x_11 +x_26 -x_28 <= 6.0
  7.0 x_10 -x_26 +x_28 <= 6.0
  7.0 x_7 -x_25 +x_26 <= 6.0
  7.0 x_6 +x_25 -x_26 <= 6.0
  7.0 x_14 +x_26 -x_30 <= 6.0
  7.0 x_15 -x_26 +x_30 <= 6.0
  1.0 <= x_5 +x_16 +x_18 <= 1.0
  1.0 <= x_4 +x_17 +x_19 <= 1.0
  7.0 x_18 -x_29 +x_31 <= 6.0
  7.0 x_19 +x_29 -x_31 <= 6.0
  7.0 x_16 -x_27 +x_31 <= 6.0
  7.0 x_17 +x_27 -x_31 <= 6.0
  1.0 <= x_9 +x_17 +x_20 <= 1.0
  1.0 <= x_8 +x_16 +x_21 <= 1.0
  7.0 x_9 -x_25 +x_27 <= 6.0
  7.0 x_8 +x_25 -x_27 <= 6.0
  7.0 x_20 +x_27 -x_30 <= 6.0
  7.0 x_21 -x_27 +x_30 <= 6.0
  7.0 x_17 +x_27 -x_31 <= 6.0
  7.0 x_16 -x_27 +x_31 <= 6.0
  1.0 <= x_13 +x_19 +x_22 <= 1.0
  1.0 <= x_12 +x_18 +x_23 <= 1.0
  7.0 x_13 -x_28 +x_29 <= 6.0
  7.0 x_12 +x_28 -x_29 <= 6.0
  7.0 x_19 +x_29 -x_31 <= 6.0
  7.0 x_18 -x_29 +x_31 <= 6.0
  7.0 x_22 +x_29 -x_30 <= 6.0
  7.0 x_23 -x_29 +x_30 <= 6.0
  1.0 <= x_15 +x_21 +x_23 <= 1.0
  1.0 <= x_14 +x_20 +x_22 <= 1.0
  7.0 x_15 -x_26 +x_30 <= 6.0
  7.0 x_14 +x_26 -x_30 <= 6.0
  7.0 x_21 -x_27 +x_30 <= 6.0
  7.0 x_20 +x_27 -x_30 <= 6.0
  7.0 x_23 -x_29 +x_30 <= 6.0
  7.0 x_22 +x_29 -x_30 <= 6.0
Variables:
  x_0 is a boolean variable (min=0.0, max=1.0)
  x_1 is a boolean variable (min=0.0, max=1.0)
  x_2 is a boolean variable (min=0.0, max=1.0)
  x_3 is a boolean variable (min=0.0, max=1.0)
```

```
x_4 is a boolean variable (min=0.0, max=1.0)
x_5 is a boolean variable (min=0.0, max=1.0)
x_6 is a boolean variable (min=0.0, max=1.0)
x_7 is a boolean variable (min=0.0, max=1.0)
x_8 is a boolean variable (min=0.0, max=1.0)
x_9 is a boolean variable (min=0.0, max=1.0)
x_10 is a boolean variable (min=0.0, max=1.0)
x_11 is a boolean variable (min=0.0, max=1.0)
x_12 is a boolean variable (min=0.0, max=1.0)
x_13 is a boolean variable (min=0.0, max=1.0)
x_14 is a boolean variable (min=0.0, max=1.0)
x_15 is a boolean variable (min=0.0, max=1.0)
x_16 is a boolean variable (min=0.0, max=1.0)
x_17 is a boolean variable (min=0.0, max=1.0)
x_18 is a boolean variable (min=0.0, max=1.0)
x_19 is a boolean variable (min=0.0, max=1.0)
x_20 is a boolean variable (min=0.0, max=1.0)
x_21 is a boolean variable (min=0.0, max=1.0)
x_22 is a boolean variable (min=0.0, max=1.0)
x_23 is a boolean variable (min=0.0, max=1.0)
x_24 is a continuous variable (min=-oo, max=+oo)
x_25 is a continuous variable (min=-oo, max=+oo)
x_26 is a continuous variable (min=-oo, max=+oo)
x_27 is a continuous variable (min=-oo, max=+oo)
x_28 is a continuous variable (min=-oo, max=+oo)
x_29 is a continuous variable (min=-oo, max=+oo)
x_30 is a continuous variable (min=-oo, max=+oo)
x_31 is a continuous variable (min=-oo, max=+oo)
```

We solve and look at the tour.

```
p.solve();
vals = p.get_values(x);
H = DiGraph();
H.set_pos(plt);

p.solve();
for u,v,l in G.edges():
    if vals[ edge_labels[ (u,v) ] ] > 0:
        H.add_edge(u,v);
    if vals[ edge_labels[ (v,u) ] ] > 0:
        H.add_edge(v,u);
H.show();
```


000
001
010
011
110
111
100
101