IEOR 4004

Due date: 19. Nov, 2019

No. / / Date. / /

HW#4

Zihui Zhan

1. $\boxed{\text{label-correcting algorithm}}$

* could handle edge-weight that are neg

* assume no negative-weight directed cycle

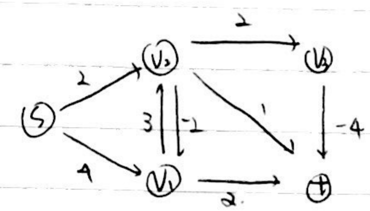
1. Initialization: $d_s=0$, $d_u=+\infty$, $pred_u$ undefined

2. main body: * if $d_v > d_u + c(u,v)$, then we reset $d_v = d_u + c(u,v)$

check arch consistency: $pred_v = u$

* else, consistent

1)



* fixed arc order: $(s,v_2)(s,v_1)(v_1,v_2)(v_2,v_1)(v_1,t)(v_2,t)(v_2,v_3)(v_3,t)$

* iteration 1:

	s	v_1	v_2	v_3	t
v_j	0	∞	∞	∞	∞
pred(j)	-	-	-	-	-

$\Rightarrow$

	s	v_1	v_2	v_3	t
v_j	0	0	2	4	0
pred(j)	-	v_2	s	v_2	v_3

* (s,v_2): $v_2 = \infty > s + c(s,v_2) = 0+2=2$

set $v_2=2$, $pred(v_2)=s$

* (s,v_1): $v_1 = \infty > s + c(s,v_1) = 0+4=4$

set $v_1=4$, $pred(v_1)=s$

* (v_1,v_2): $v_2 = 2 < v_1 + c(v_1,v_2) = 4+3=7$

consistent $\Rightarrow$ unchanged

* (v_2,v_1): $v_1 = 4 > v_2 + c(v_2,v_1) = 2-2=0$

set $v_1 = 0$, $pred(v_1)=v_2$

* (v_1,t): $t = \infty > v_1 + c(v_1,t) = 0+2=2$

set $t=2$, $pred(t)=v_1$

* (v_2,t): $t=2 < v_2 + c(v_2,t) = 2+1=3$

consistent

* (v_2,v_3): $v_3 = \infty > v_2 + c(v_2,v_3) = 2+2=4$

set $v_3=4$, $pred(v_3)=v_2$

* (v_3,t): $t = 2 > v_3 + c(v_3,t) = 4-4=0$

set $t=0$, $pred(t)=v_3$

* Iteration 2

(S, V_2): $V_2 = 2 = S + c(S, V_2) = 2$

consistent → unchanged

(S, V_1): $V_1 = 0 < S + c(S, V_1) = 4$

(V_1, V_2): $V_2 = 2 < V_1 + c(V_1, V_2) = 0+3 = 3$

(V_2, V_1): $V_1 = 0 = V_2 + c(V_2, V_1) = 2-2$

(V_1, t): $t = 0 < V_1 + c(V_1, t) = 0+2$

(V_2, t): $t = 0 < V_2 + c(V_2, t) = 2+1$

(V_2, V_3): $V_3 = 4 = V_2 + c(V_2, V_3) = 2+2$

(V_3, t): $t = 0 = V_3 + c(V_3, t) = 4-4 = 0$

Thus, all arcs are consistent in the 2nd iteration

shortest path: $S - V_2 - V_3 - t$ with weight = 0

2) final table

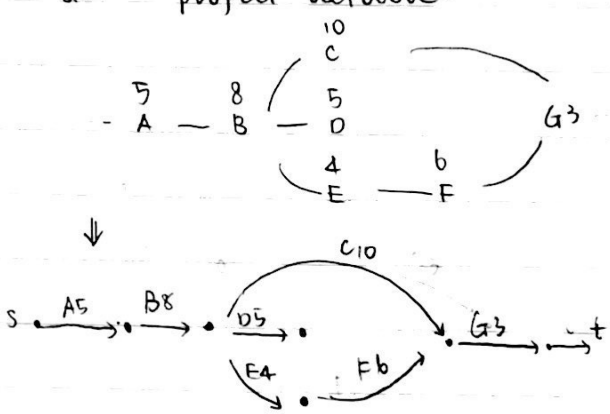
	S	V_1	V_2	V_3	t
V_j	0	0	2	4	0
pred(V_j)	-	V_2	S	V_2	V_3

*

shortest path from v_1 to v_3: unknown from the final table.

shortest path from v_2 to t: $V_2 - V_3 - t$

* 2. project network.



through eyeballing,

there are two critical paths

* S - A - B - C - G - t

* S - A - B - E - F - G - t

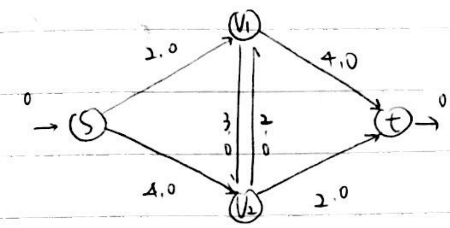
with weight 26

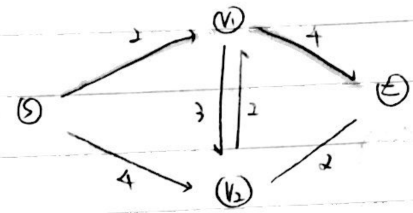
* feasible flow: arc上可以(已经)所通过的flow

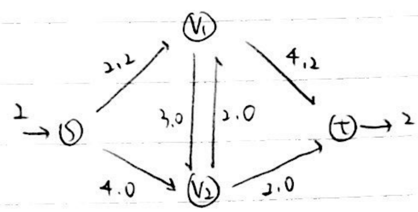
No. / / Date. / /

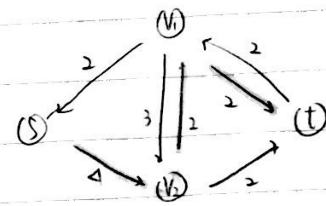
Residual Network

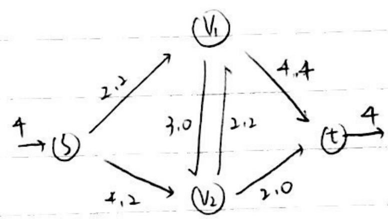
3.

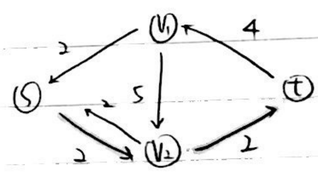




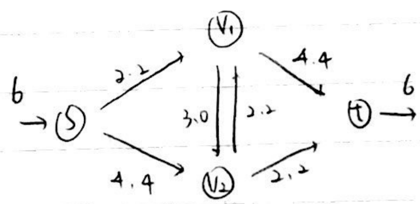


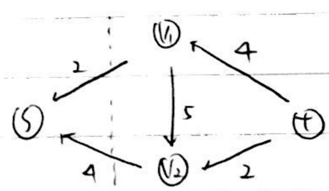






*





the maximum flow is 6 (from s to t)

minimum cut $S = \{S\}$ $\bar{S} = \{V_1, V_2, t\}$

4. No, we can't

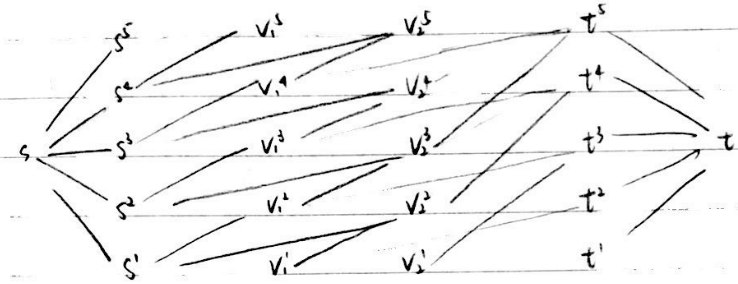
* It has 2 s-t flow of max value: $s-V_1-V_2-t$ & $s-V_1-V_3-t$

* However, the min cut $S = \{S\}$, $\bar{S} = \{V_1, V_2, V_3, t\}$ with capacity 1

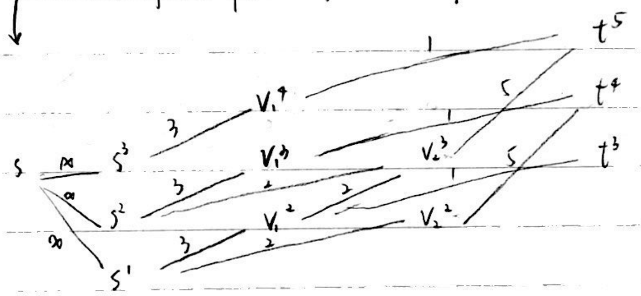
and there's only one such case.

5. If we don't need to increase by an integer each time, then the number of iteration is infinite. For example, we increase the flow of residual network *wording by choosing s-$v_i \cdots v_j$-t path with capacity ε. Then, by algorithm, we only increase the flow by $\frac{\varepsilon}{2}$ instead of ε. $\Rightarrow$ the s-$v_i \cdots v_j$-t path still has capacity $\frac{\varepsilon}{2}$. $\Rightarrow$ we increase flow of the path by $\frac{\varepsilon}{4}$ $\Rightarrow$ for each iteration n, we may still increase the flow by $\frac{\varepsilon}{2^{n+1}}$ in the next round

6. 1)



↓ * all flows from left to right



2) maximum flow over time: is 9

t^3: $s^1 - v_1^2 - t^3$: 1

t^4: $s^1 - v_2^2 - t^4$: 2

$s^2 - v_1^3 - t^4$: 1 } 3

t^5: $s^3 - v_1^4 - t^5$: 1

$s^1 - v_1^2 - v_2^3 - t^5$: 2

$s^2 - v_2^3 - t^5$: 2 } 5

} 9

3). (v_2, t) at $t = 3$-7. 6 units of flows are traversing

No. / / Date. / / /

7. 2) let ij denotes the amount of money that bank i transfers to bank j

Minimize

object: $3\times(ab+ac+da+ae+cf+ag+bc+db+be+fg+gb+cd+ce+fc+cg$
$+de+fd+dg+fe+ge+fg+ba+ca+ad+ea+fa+ga+cb+bd+eb$
$+bf+bg+dc+ec+cf+gc+ed+df+gd+ef+eg+gf)$

Constraints: $ab+ac+ad+ae+af+ag-(ba+ca+da+ea+fa+ga)=62$

$ba+bc+bd+be+bf+bg-(ab+cb+db+eb+fb+gb)=-117$

$ca+cb+cd+ce+cf+cg-(ac+bc+dc+ec+fc+gc)=81$

$da+db+dc+de+df+dg-(ad+bd+cd+ed+fd+gd)=145$

$ea+eb+ec+ed+ef+eg-(ae+be+ce+de+fe+ge)=-128$

$fa+fb+fc+fd+fe+fg-(af+bf+cf+df+ef+gf)=105$

$ga+gb+gc+gd+ge+gf-(ag+bg+cg+dg+eg+fg)=-148$

$ij\geq 0 \quad \forall\ i,j\in\{a,b,c,d,e,f,g\}$

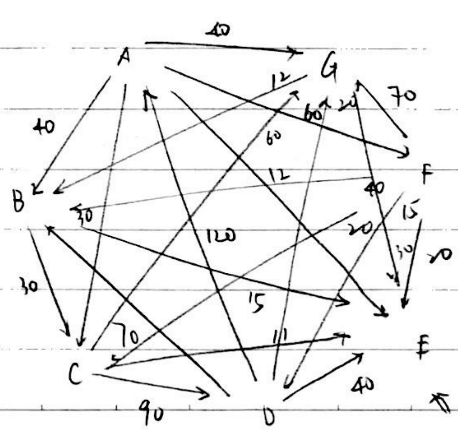
Using Gurobi to solve the system, ~~the~~ I achieved the optimal objective as 1179 dollars

1). min-cost u_{ij}= capacity of edge $(i,j)\in E$

ij= payment from bank (node) i to bank (node) j

* graph.

min-cost



Network $G=(V,E)$

obj: $\min 3\cdot\sum_i\sum_j ij$ where $i\neq j$.

Since each edge is in thousand of dollars

transaction fee $=0.003\times1000=3\$$

Code

November 24, 2019

[1]:
```
from gurobipy import *


# create a model
m = Model()

# create variables
ab = m.addVar(vtype=GRB.INTEGER, name="ab", lb=0)
ac = m.addVar(vtype=GRB.INTEGER, name="ac", lb=0)
da = m.addVar(vtype=GRB.INTEGER, name="da", lb=0)
ae = m.addVar(vtype=GRB.INTEGER, name="ae", lb=0)
af = m.addVar(vtype=GRB.INTEGER, name="af", lb=0)
ag = m.addVar(vtype=GRB.INTEGER, name="ag", lb=0)
bc = m.addVar(vtype=GRB.INTEGER, name="bc", lb=0)
db = m.addVar(vtype=GRB.INTEGER, name="db", lb=0)
be = m.addVar(vtype=GRB.INTEGER, name="be", lb=0)
fb = m.addVar(vtype=GRB.INTEGER, name="fb", lb=0)
gb = m.addVar(vtype=GRB.INTEGER, name="gb", lb=0)
cd = m.addVar(vtype=GRB.INTEGER, name="cd", lb=0)
ce = m.addVar(vtype=GRB.INTEGER, name="ce", lb=0)
fc = m.addVar(vtype=GRB.INTEGER, name="fc", lb=0)
cg = m.addVar(vtype=GRB.INTEGER, name="cg", lb=0)
de = m.addVar(vtype=GRB.INTEGER, name="de", lb=0)
fd = m.addVar(vtype=GRB.INTEGER, name="fd", lb=0)
dg = m.addVar(vtype=GRB.INTEGER, name="dg", lb=0)
fe = m.addVar(vtype=GRB.INTEGER, name="fe", lb=0)
ge = m.addVar(vtype=GRB.INTEGER, name="ge", lb=0)
fg = m.addVar(vtype=GRB.INTEGER, name="fg", lb=0)

ba = m.addVar(vtype=GRB.INTEGER, name="ba", lb=0)
ca = m.addVar(vtype=GRB.INTEGER, name="ca", lb=0)
ad = m.addVar(vtype=GRB.INTEGER, name="ad", lb=0)
ea = m.addVar(vtype=GRB.INTEGER, name="ea", lb=0)
fa = m.addVar(vtype=GRB.INTEGER, name="fa", lb=0)
ga = m.addVar(vtype=GRB.INTEGER, name="ga", lb=0)
cb = m.addVar(vtype=GRB.INTEGER, name="cb", lb=0)
bd = m.addVar(vtype=GRB.INTEGER, name="bd", lb=0)
```

```
eb = m.addVar(vtype=GRB.INTEGER, name="eb", lb=0)
bf = m.addVar(vtype=GRB.INTEGER, name="bf", lb=0)
bg = m.addVar(vtype=GRB.INTEGER, name="bg", lb=0)
dc = m.addVar(vtype=GRB.INTEGER, name="dc", lb=0)
ec = m.addVar(vtype=GRB.INTEGER, name="ec", lb=0)
cf = m.addVar(vtype=GRB.INTEGER, name="cf", lb=0)
gc = m.addVar(vtype=GRB.INTEGER, name="gc", lb=0)
ed = m.addVar(vtype=GRB.INTEGER, name="ed", lb=0)
df = m.addVar(vtype=GRB.INTEGER, name="df", lb=0)
gd = m.addVar(vtype=GRB.INTEGER, name="gd", lb=0)
ef = m.addVar(vtype=GRB.INTEGER, name="ef", lb=0)
eg = m.addVar(vtype=GRB.INTEGER, name="eg", lb=0)
gf = m.addVar(vtype=GRB.INTEGER, name="gf", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
    3*(ab + ac + da + ae + af + ag + bc + db + be + fb + gb + cd + ce + fc + cg␣
 ↪+ de + fd + dg + fe + ge +
       fg + ba + ca + ad + ea + fa + ga + cb + bd + eb + bf + bg + dc + ec + cf␣
 ↪+ gc + ed + df + gd + ef + eg + gf),
    GRB.MINIMIZE
)

# add constraints
m.addConstr(ab + ac + ad + ae + af + ag - (ba + ca + da + ea + fa + ga) == 62)
m.addConstr(ba + bc + bd + be + bf + bg - (ab + cb + db + eb + fb + gb) == -117)
m.addConstr(ca + cb + cd + ce + cf + cg - (ac + bc + dc + ec + fc + gc) == 81)
m.addConstr(da + db + dc + de + df + dg - (ad + bd + cd + ed + fd + gd) == 145)
m.addConstr(ea + eb + ec + ed + ef + eg - (ae + be + ce + de + fe + ge) == -128)
m.addConstr(fa + fb + fc + fd + fe + fg - (af + bf + cf + df + ef + gf) == 105)
m.addConstr(ga + gb + gc + gd + ge + gf - (ag + bg + cg + dg + eg + fg) == -148)

# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

```
Academic license - for non-commercial use only
Optimize a model with 7 rows, 42 columns and 84 nonzeros
Variable types: 0 continuous, 42 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [3e+00, 3e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [6e+01, 1e+02]
Found heuristic solution: objective 2370.0000000
Presolve time: 0.02s
Presolved: 7 rows, 42 columns, 84 nonzeros
Variable types: 0 continuous, 42 integer (0 binary)

Root relaxation: objective 1.179000e+03, 6 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    1179.0000000 1179.00000  0.00%     -    0s

Explored 0 nodes (6 simplex iterations) in 0.13 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 1179 2370

Optimal solution found (tolerance 1.00e-04)
Best objective 1.179000000000e+03, best bound 1.179000000000e+03, gap 0.0000%
Model status:  2
ab 12.0

ac -0.0

da -0.0

ae 47.0

af -0.0

ag 3.0

bc -0.0

db -0.0

be -0.0

fb 105.0
```

gb -0.0

cd -0.0

ce 81.0

fc -0.0

cg -0.0

de -0.0

fd -0.0

dg 145.0

fe -0.0

ge -0.0

fg -0.0

ba -0.0

ca -0.0

ad -0.0

ea -0.0

fa -0.0

ga -0.0

cb -0.0

bd -0.0

eb -0.0

bf -0.0

bg -0.0

dc -0.0

ec -0.0

```
cf -0.0

gc -0.0

ed -0.0

df -0.0

gd -0.0

ef -0.0

eg -0.0

gf -0.0

---------------
Obj Value:  1179.0
```

[ ]: