$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM*

* Algorithm actually takes $\tilde{O}(m^{4/3})$ time but the tighter analysis would take 2 lectures.

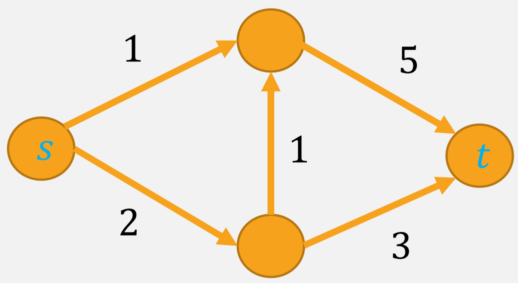
AGENDA

- Max-flow
- Physics (Electrical Flows, Ohm's Law, Kirchhoff Law) Review (30 minutes)
- Properties of Electrical Networks & Applications
- Fast Laplacian Solvers using Johnson-Lindenstrauss Theorem
- Multiplicative Weight Update MWU Review (30 mins)
- $\tilde{O}(m^{1.5})$ max-flow Algorithm. (10 mins)

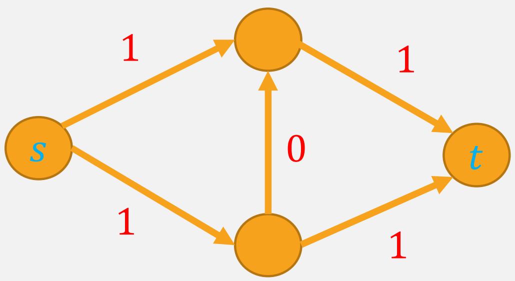
MAX-FLOW

- Given a directed graph $G(V, E)$ with edge capacities $c(e)$, and two distinguished vertices s, t, find the **maximum flow** from s to t
- A flow is an assignment $f: E \to \Re_+$ that satisfies:
  - $f(e) \leq c(e)$ for all $e \in E$ (Capacity constraint)
  - $\sum_{u:(u,v)\in E} f(u,v) = \sum_{u:(v,u)\in E} f(v,u)$ for all $u, v \in V - \{s,t\}$ (Conservation of flow)
- Flow value is $\sum_{u\in V} f(s,u)$ (Total flow going out of s), or into t

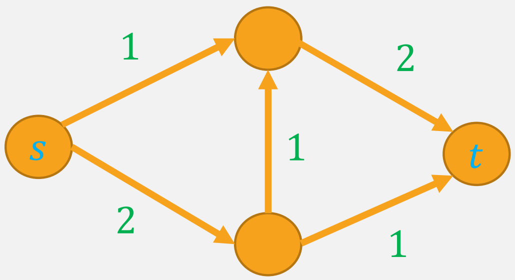
EXAMPLE



Graph and Capacities



Flow



Max-Flow

PHYSICS & LINEAR ALGEBRA REVIEW

- Consider an undirected graph $G(V, E)$ such that each edge (i, j) has resistance $r(i, j)$ (or conductance $c(i, j) = \frac{1}{r(i,j)}$).
- A **current flow** $f(i, j)$ is one that obeys both:
  - Kirchhoff's current law:

    Flow into node v = flow leaving node v
  - Ohm's Law:

    There exists a potential $p(v)$ such that $f(i, j) = \frac{p(i) - p(j)}{r(i,j)}$ for all $i, j \in V$.

    Note p is translation invariant.

    $p(i) - p(j)$ acts like "Voltage", $f(i, j)$ as current, and $r(i, j)$ as resistance. $\left(\boldsymbol{I} = \frac{\boldsymbol{V}}{\boldsymbol{R}}\right)$

OHM LAW - CONTD

Ohm's Law:

There exists a potential $p(v)$ such that $f(i,j) = \frac{p(i)-p(j)}{r(i,j)}$ for all $i, j \in V$.

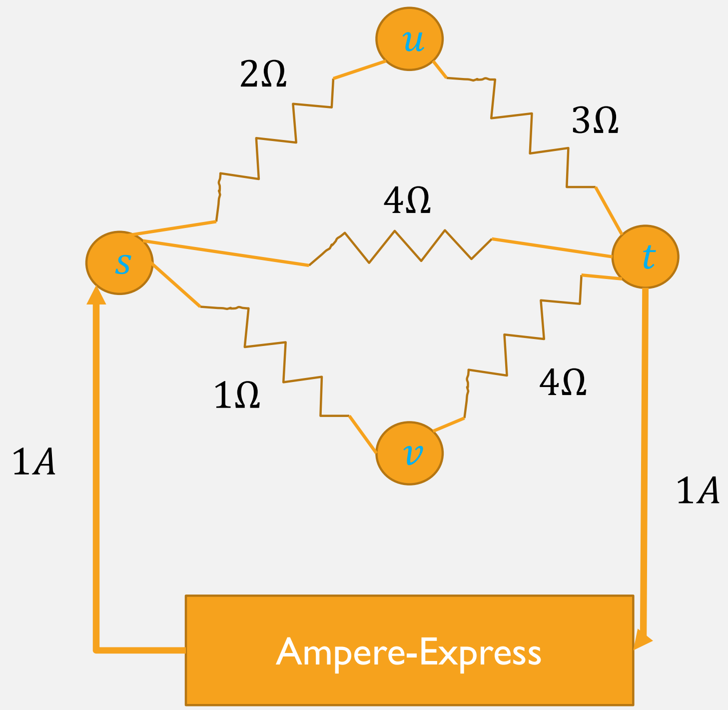
$p = 0, f = 0$ satisfies Ohm's law.

Define $b(u) = \sum_v f(u,v)$ as the total flow (or current) into u.

To make things more interesting, we force $b(s) = 1, b(t) = -1$. Excludes $p = 0, f = 0$.

Any such flow is an **electrical-flow**

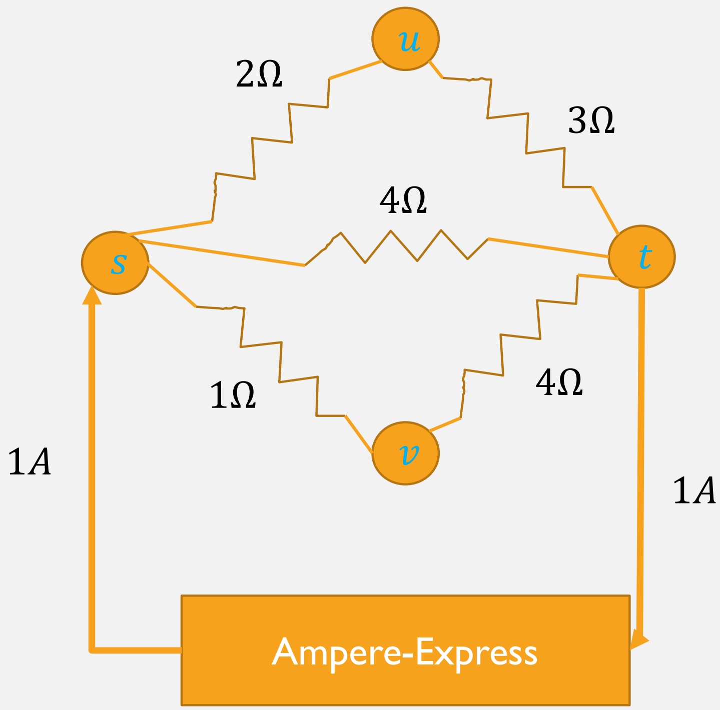
EXAMPLE



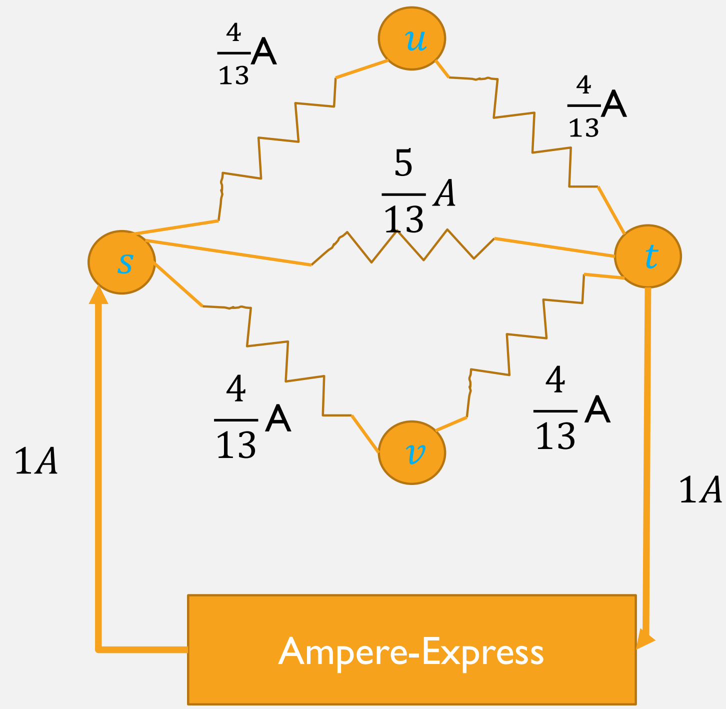
How do we find the potentials?

Ohm's Law!

EXAMPLE

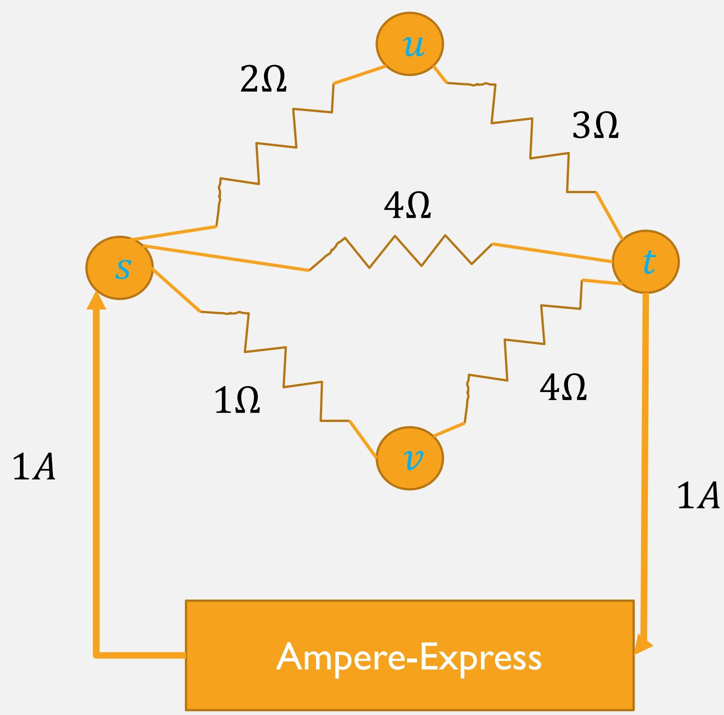


u
2Ω
3Ω
4Ω
s
t
1Ω
4Ω
v
1A
1A
Ampere-Express

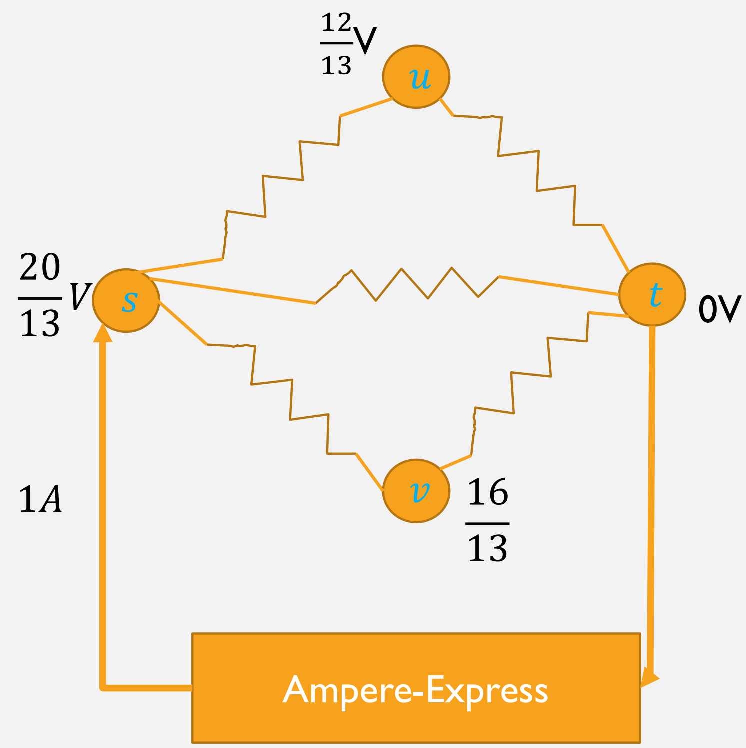


4/13 A
u
4/13 A
5/13 A
s
t
4/13 A
4/13 A
v
1A
1A
Ampere-Express

EXAMPLE



2Ω
3Ω
4Ω
u
s
t
v
1Ω
4Ω
1A
1A
Ampere-Express



$\frac{12}{13}$V
u
$\frac{20}{13}V$
s
t
0V
v
$\frac{16}{13}$
1A
Ampere-Express

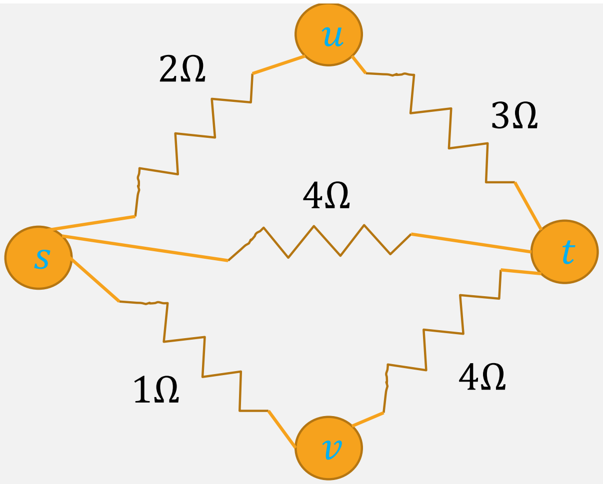
TANGENT 1 - LAPLACIANS

- Consider an undirected graph $G(V, E)$ with adjacency matrix A (which can be weighted). Let D be the (diagonal) degree matrix defined as $d_{ii} = \deg_G(i) = \sum_j a_{ij}$ and 0 otherwise.
- The Laplacian matrix is defined as $L_G = D - A$. VERY useful in Spectral Graph Theory.

- Breakthrough result of Teng et al. from 2004:
  - Given a system of equations $L_{m\times n}\, x = b$ where L is a diagonally dominant matrix, one can find an approximate solution $\hat{x}$ in $\tilde{O}(m)$ time.
  - Specifically, one can approximately compute $L^+ b$ where L^+ is the pseudoinverse of L for diagonally dominant matrices.
  - A diagonally dominant matrix A is a matrix satisfying $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$ for all i.
  - One can prove that the Laplacian matrix is diagonally dominant.
  - Needs a whole lecture for itself…

OHM & KIRCHOFF'S LAWS CONTD

- Combining both laws and some linear algebra magic, we can find a necessary condition for potentials and current.
- Suppose for a given b (recall $b(u) = \sum_v f(u, v)$) that we want to find the corresponding potentials $p(u)$ for the resulting **electrical flow**.
- Then $L_G p = b$, where the weights in the adjacency matrix are $\frac{1}{r(i,j)} = c(i, j)$.
- Intuitively, and with lots of handwaving, recall that "$V = IR$" and $p = L_G^+ b$. "b" acts as current I. L_G^+ acts like L_G^{-1}, which we want to be "resistance".



$b =$

s	1
u	0
v	0
t	-1

$L_G = D - A$ and $L_G p = b$

$p = L_G^+ b =$

s	0.62
u	0
v	0.31
t	-0.92

$A =$

	s	u	v	t
s	0	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{1}{4}$
u	$\frac{1}{2}$	0	0	$\frac{1}{3}$
v	1	0	0	$\frac{1}{4}$
t	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{4}$	0

$D =$

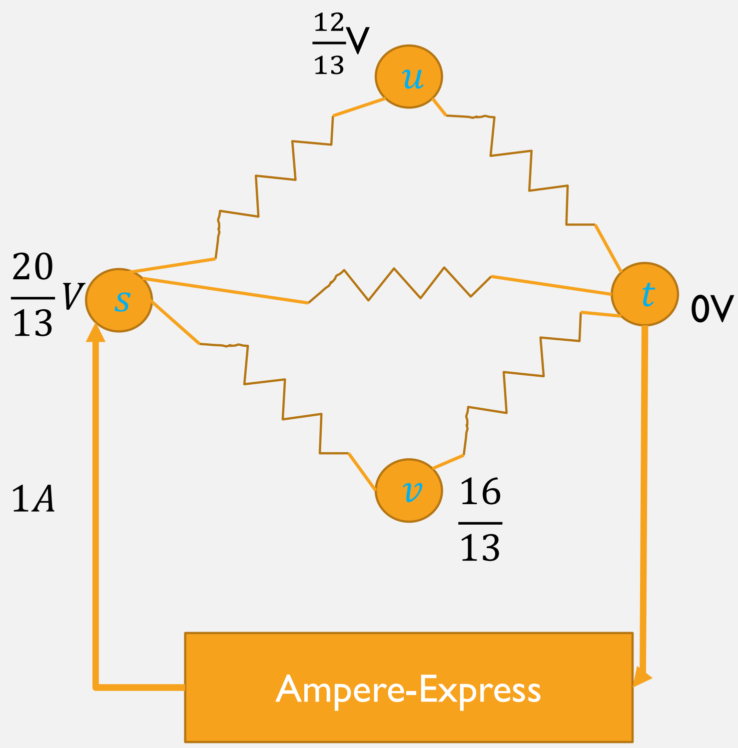
	s	u	v	t
s	$\frac{7}{4}$	0	0	0
u	0	$\frac{5}{6}$	0	0
v	0	0	$\frac{5}{4}$	0
t	0	0	0	$\frac{5}{6}$

p Is Translation Invariant:

$p =$

s	1.538
u	0.923
v	1.231
t	0

EXAMPLE



$p =$

s	1.538
u	0.923
v	1.231
t	0

OHM & KIRCHOFF'S LAWS REVISITED

- So to find the potentials that induce demands b, we only need to solve one system:

$$L_G p = b$$

- Using Teng's result, can solve it in $\tilde{O}(m)$ time for one b!

EFFECTIVE RESISTANCE

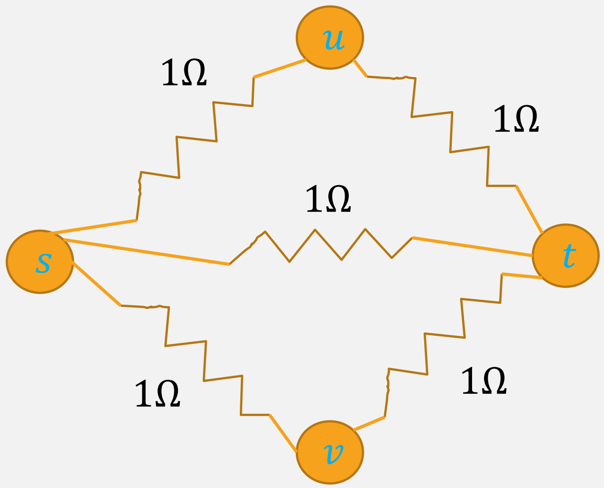
- Effective resistance is the potential drop between two adjacent vertices assuming we push one unit of current into one and out of the other.

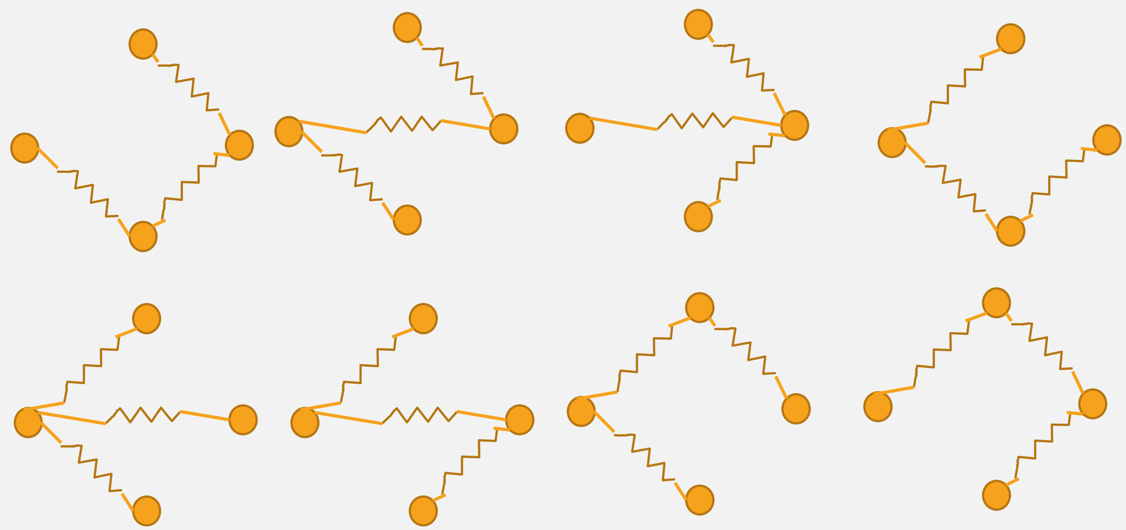
- More formally, $r_{eff}(i,j) = p(i) - p(j)$ for $ij \in E$ when the demands are $b_i = 1, b_j = -1$.

- We can compute it for each edge by solving $L_G p = b^{ij}$ where $b_i^{ij} = 1, b_j^{ij} = -1$ and 0 otherwise.

- Here is something to blow your mind.

TANGENT 2 – NUMBER OF TREES CONTAINING AN EDGE





$b =$

s	1
u	0
v	0
t	-1

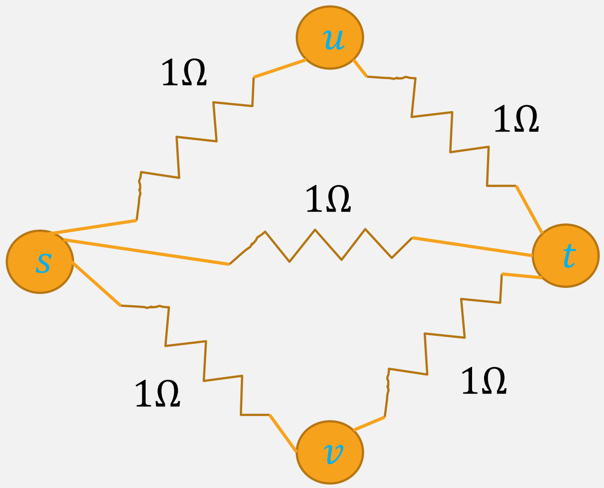
$p =$

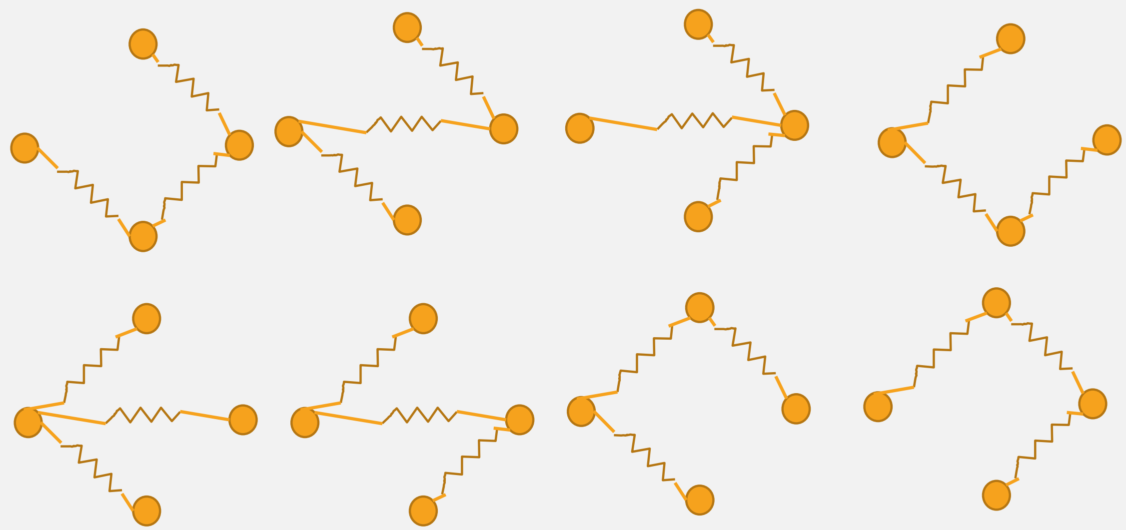
s	0.5
u	0.25
v	0.25
t	0

The effective resistance between st is $p(s) - \mathrm{p}(\mathrm{t}) = 0.5$.

The edge st appears in 4/8 of the spanning trees of G!

TANGENT 2 – NUMBER OF TREES CONTAINING AN EDGE





$b =$

s	1
u	-1
v	0
t	0

$p =$

s	0.625
u	0
v	0.5
t	0.375

The effective resistance between su is $p(s) - p(\mathrm{u}) = 0.625$.

The edge su appears in $5/8 = 0.625$ of the spanning trees of G!

TANGENT 2 – NUMBER OF TREES CONTAINING AN EDGE

The coolest Theorem you'll see this week:

Let $G(V, E)$ be an undirected graph. If we uniformly sample a random spanning tree from G, then the probability that $ij \in T$ is $r_{eff}(i, j)$ in the corresponding resistor network !

This is based in real physics! You can set up a resistance network to find the probabilities an edge is in a random spanning tree with an amperometer!

Best known algorithm to compute it runs in $\tilde{O}(\frac{m}{\epsilon^2})$ due to Chandra and Kent in SODA 21 based on blocking flows.

Physics evidence suggests there might be linear time algorithms.

COMPUTING EFFECTIVE RESISTANCE

- Effective resistance is the potential drop between two adjacent vertices. More formally, $r_{eff}(i,j) = p(i) - p(j)$ for $ij \in E$
- For each edge $ij \in E$, define $\mathrm{b}^{ij} \in \Re^n$ such that $b_i^{ij} = 1, b_j^{ij} = -1$.
- To compute the effective resistance for all edges, we can solve $L_G p = b^{ij} \ \ \forall ij \in E$. Takes $\tilde{O}(m^2)$ time.
- However, we can approximate all effective resistances in $\tilde{O}(m)$!
- Recall $p = L_G^{+} b$, and so $r_{eff}(i,j) = \left(e_i - e_j\right)^T L_G^{+}(e_i - e_j)$. Notice that
  - $r_{eff}(i,j) = \left\| L_G^{\frac{+}{2}}\left(e_i - e_j\right) \right\|_2 = \left\| v_i - v_j \right\|_2$ Where $v_i = L_G^{\frac{+}{2}} e_i$.
  - Note: If L is diagonally dominant, then $L^{1/2} x = b$ can still be solved using Teng's Method, so we're still Kosher.
  - Since effective resistance are L_2 distances, we can use Johnson-Lindenstrauss lemma to approximate them using vectors v_i. Details omitted.

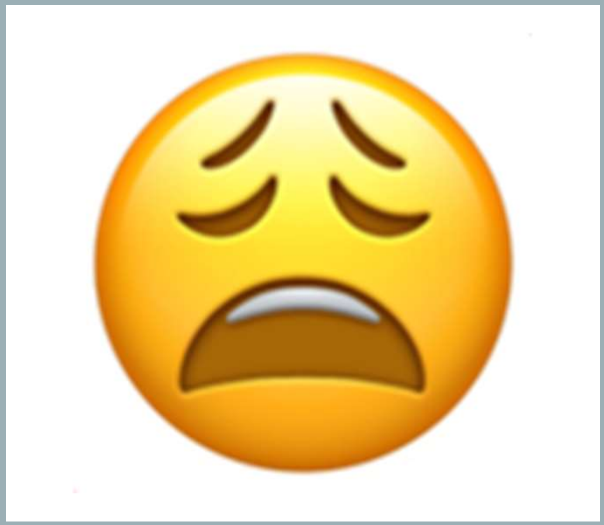
COMPUTING EFFECTIVE RESISTANCE

- One last interesting property about electrical flows.
- Nature is efficient, and so if we look at the energy dissipated in the network between s, t, it turns out electrical flows minimizes that. In particular:

$$\sum_{e \in E} r_e f(e)^2$$

Is minimized by electrical flows. (Recall $Power = I^2 R$)

Proof uses Linear algebra, not very insightful. "Common" physics knowledge.



END OF PHYSICS REVIEW

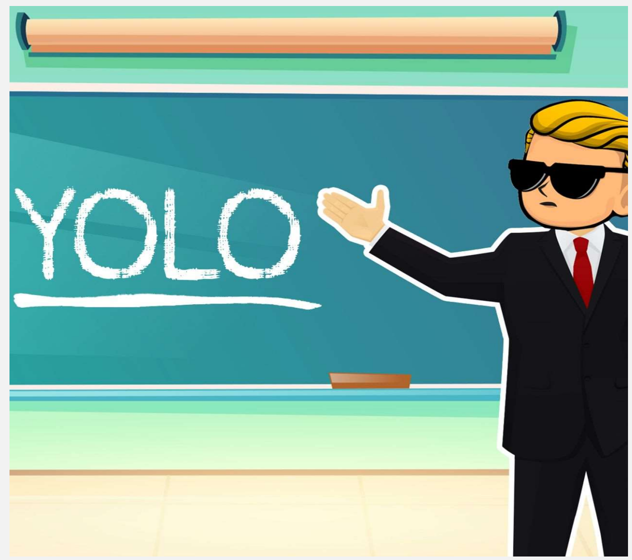
MULTIPLICATIVE WEIGHT UPDATES (MWU)

WHAT IS MWU?

- MWU is a "meta" algorithm, in the same sense of gradient descent. In fact, it generalizes gradient descent and many known optimization algorithms.
- Extremely useful in optimization.

WHAT IS MWU?

- Want to bet on AMC stock. Have 3 fine experts to lean on:



- You have no idea who is legit and who isn't. (Hint: hint)
- You want to bet everyday on 0DTE expiration options… Cause YOLO.
- Each expert either says STONKS 📈 or NOT STONKS 📉 everyday. Based on recommendations, you need to make a decision

WHAT IS MWU?

- Initialize trust weights as 1 for all:
- Update rule is:

$w_i^{t+1} = (1-\epsilon)w_i^t$ If "expert" answered incorrectly.

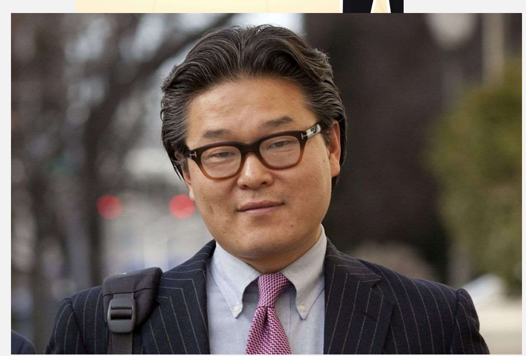
$w_i^{t+1} = \mathrm{w}_\mathrm{i}^t$ If "expert" answered correctly.

Our bet is 📈 if the total weight of all experts predicting up 📈 at least $\sum_i w_i^t/2$ and 📉 otherwise.

Fix $\epsilon = 0.1$ for example.

1

1



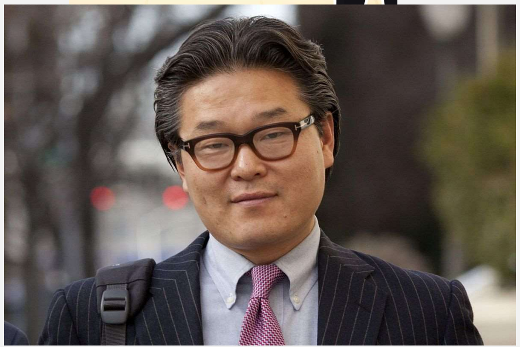
1

WHAT IS MWU?

- Day 1 Guess:
- Day 1 Result:
- Weight updates:

1

1



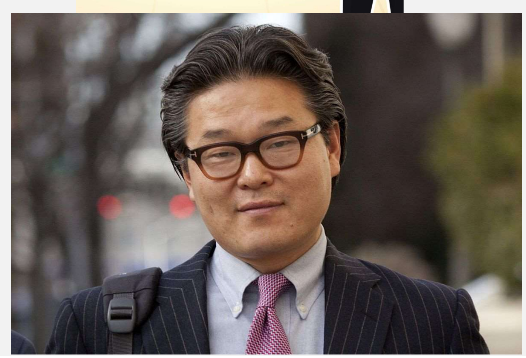
1

WHAT IS MWU?

- Day 1 Guess:
- Day 1 Result:
- Weight updates:

0.9

0.9



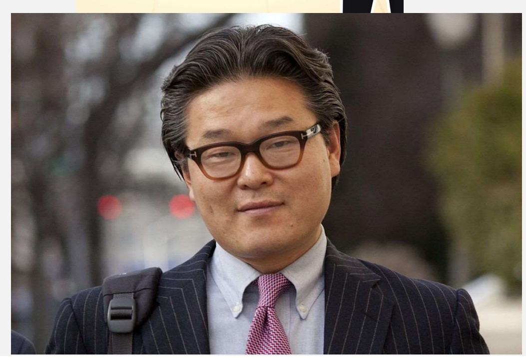
1

WHAT IS MWU?

- Day 2 Guess:
- Day 2 Result:
- Weight updates:

0.9

0.9



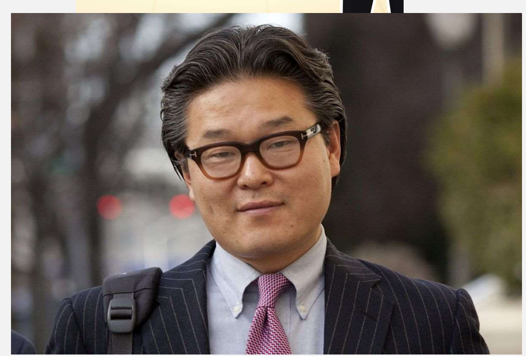
1

WHAT IS MWU?

- Day 2 Guess:
- Day 2 Result:
- Weight updates:

0.81

0.9



0.9

WHAT IS MWU?

- Theorem: Let m_i^t be the number of mistakes that ”expert” $i, 1 \leq i \leq n$, does after t days. Let M_t be the number of mistakes we make after t days. Then

$$M_t \leq \frac{2 \log n}{\epsilon} + (1 + \epsilon) m_i^t \qquad \text{For ALL experts } i!$$

WHAT IS MWU?

$$M_t \leq \frac{2\log}{\epsilon} + 2(1+\epsilon)m_i^t$$ For ALL experts i!

Pf: Define the potential function $\Phi^t = \sum_i w_i^t$ with $\Phi^1 = n$.

Every time we are wrong, at least half the weight decreases by $(1-\epsilon)$ factor. So

$$\Phi^{t+1} \leq \Phi^t\left(\frac{1}{2} + \frac{1}{2}(1-\epsilon)\right) = \Phi^t(1-\frac{\epsilon}{2})$$

Solving the recurrence, we get

$$\Phi^t \leq n\left(1-\frac{\epsilon}{2}\right)^{M_t}$$

But

$$\Phi^t \geq w_i^t \qquad \forall i$$

Rearranging, and approximating $-\log(1-x) \leq x + x^2$ For $x \leq \frac{1}{2}$ yields the results.

WHAT IS MWU?

Generalized Algorithm for optimization.

We assume there is a matrix M such that $M(i,j)$ is the penalty that expert i pays when the outcome is $j \in \boldsymbol{P}$ where $\boldsymbol{P}$ is set of outcomes**.**

Assume $M(i,j) \in [0,\ \rho]$. We call ρ the **width** of the oracle M

Every step t we have trust scores w_i^t to expert i.

We associate time t with distribution $D^t = \{p_1^t, \dots, p_n^t\} = \{\frac{w_1^t}{\sum w_i^t}, \dots, \frac{w_n^t}{\sum w_i^t}\}$

We pick an expert according to distribution D^t, and use it to make our prediction. Based on the outcome j_t in round t, the weight is updated:

$$w_i^{t+1} = w_i^t(1 - \frac{\epsilon M(i,j_t)}{\rho})$$

WHAT IS MWU?

Theorem: After T rounds, for any expert i, we have

$$\sum_t \sum_i p_i^t w_i^t \leq \frac{\rho \log(n)}{\epsilon} + (1+\epsilon) \sum_t M(i, j_t)$$

Almost identical potential proof to theorem from before.

WHAT IS MWU?

Figuring out what the "experts" are, what the penalties/rewards $M(i,j)$ are, and what "outcomes" is the hardest part of using MWU.

Sometimes requires very "clever" outlooks.

WHAT IS MWU?

Packing/Covering Linear Programs (Known as Plotkin, Shmoys, Tardos framework)

Problem: Is there $x \in P$ where $Ax \geq b$? (Feasibility).

Think of P as the "easy" constraints, and A as the hard ones.

Implicitly assumes the constraint rows of A are the "experts"

Assume the following oracle is known: $\exists ?\, x \in P : c^T x \geq (1-\epsilon) d$ where $c = \sum_i p_i^t A_i$ and $d = \sum_i p_i^t b_i$?

Natural ➔ exists for many problems.

If so, we can solve the original feasibility problem with MWU!

WHAT IS MWU?

Packing/Covering Linear Programs (Known as Plotkin, Shmoys, Tardos framework)

Problem: Is there $\boldsymbol{x \in P}$ where $\boldsymbol{Ax \geq b}$? (Feasability).

The "experts" are the **constraints**. Events correspond to **vectors** $x \in P$! The oracle penalty is $A_i x - b_i$, how badly the inequality is not satisfied.

The **width** here is $\rho = \max_{x \in P}(A_i x - b_i)$ which can be unbounded (but there is a trick to get around this).

Theorem: It takes $\widetilde{O}(\frac{\rho}{\epsilon^2})$ oracle calls for MWU to converge to x such that $Ax \geq (1-\epsilon)b$ (or conclude no such x exists).

Designing correct MWU algorithms is an art that is not easy to master. See Arora's survey for a lot more examples.

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Idea: Apply Plotkin, Shmoys, Tardos framework on the Max-Flow linear program:

$$\max\ |f| = \sum_u f(s,u)$$

$$f(e) \le c(e) \qquad \forall e \in E$$

$$\sum_v f(u,v) = \sum_v f(v,u) \quad \forall u \in V - \{s,t\}$$

$$f_e \ge 0 \qquad \forall e \in E$$

Assume $c(e) = 1$, makes presentation less messy and doesn't loose generality in proof.

Binary search on F^*, the maximum flow value.

"Easy" constraints are flow conservation, non negativity, and $|f| \ge F$. This is is the "P" from the Tardos framework.

Hard constraints are $f(e) \le c(e) = 1$. Or $I_m f \le c$. This is the "$Ax \le b$" from the Tardos framework.

What's the width here? It is is $\max_i A_i x - b_i = \max_{e \in E} f(e) - 1$

Max flow with unit capacity equivalent to: $\exists ?\ f \in P$ such that $If \le 1$.

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Idea: Apply Plotkin, Shmoys, Tardos framework on the Max-Flow linear program:

$$\begin{aligned} &\max |f| \\ f(e) &\le c(e) && \forall e \in E \\ \sum_v f(u,v) &= \sum_v f(v,u) && \forall u \in V - \{s,t\} \\ f_e &\ge 0 && \forall e \in E \end{aligned}$$

Max flow with unit capacity equivalent to: $\exists ?\, f \in P$ such that $If \le 1$.

What oracle do we need? $\exists ?\, f$ such that $|f| \ge F$ (guessed max flow value) and $f(e) \ge 0$ and conserving flow such that

$$\sum_{e \in E} p_e^t f(e) \le (1+\epsilon) \sum_{e \in E} p_e^t$$

Intuitively, this is saying the "average" capacity constraint is (approximately) satisfied.

We will answer this oracle with electrical flows!

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Max flow:

$$f(e) \le c(e) \qquad \forall e \in E$$
$$\sum_v f(u,v) = \sum_v f(v,u) \qquad \forall u \in V - \{s,t\}$$
$$f_e \ge 0 \qquad \forall e \in E$$
$$|f| = F$$

Electrical flows with resistance:

$$\sum_v f(u,v) = \sum_v f(v,u) \qquad \forall u \in V - \{s,t\}$$
$$f_e \ge 0 \qquad \forall e \in E$$
$$|f| = F$$

Only thing we can control is resistances on edges. Can we play with the resistances on edges to force

$$\sum_{e \in E} p_e^t f(e) \le (1+\epsilon) \sum_{e \in E} p_e^t$$

Intuitively, even though flow doesn't have to respect capacity, can we force it to respect it "on average"?

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Idea: Apply Plotkin, Shmoys, Tardos framework on the Max-Flow linear program:

$$\max |f|$$
$$f(e) \le c(e) \qquad \forall e \in E$$
$$\sum_v f(u,v) = \sum_v f(v,u) \qquad \forall u \in V - \{s,t\}$$
$$f_e \ge 0 \qquad \forall e \in E$$

Construct an electrical network with resistances $r_e = p_e^t + \frac{\epsilon \sum_{e \in E} p_e^t}{m}^{(*)}$. Put F units of flow into s and $-F$ units from t.

Conservation and non-negativity of flow is free. Pushes F flow from s using demands. So "easy" constraints are all good.

Just need to prove the **average capacity is respected** and **bound the width**. Then apply Tardos framework.

(*) Yes $\sum_{e \in E} p_e^t = 1$, but in the capacitated case, it should be $\sum_{e \in E} p_e^t \, c(e) \neq 1$ so I'll leave it as it is.

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Theorem: If we set $r_e = p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}$,The electrical flow oracle satisfies $\sum_{e\in E} p_e^t f(e) \leq (1+\epsilon)\sum_{e\in E} p_e^t$ and in addition, the width is $\rho = O(\sqrt{\frac{m}{\epsilon}})$

Proof: Let f be the optimal electrical flow. We have

$$\sum_{e\in E} p_e^t f(e) \leq \sqrt{\sum_{e\in E} p_e^t f(e)^2}\sqrt{\sum_{e\in E} p_e^t} \qquad (1)$$

So it suffices to show

$$\sum_{e\in E} p_e^t f(e)^2 \leq (1+\epsilon)\sum_{e\in E} p_e^t \qquad (2)$$

Because then (1) becomes:

$$\sum_{e\in E} p_e^t f(e)) \leq \sqrt{1+\epsilon}\sum_{e\in E} p_e^t$$

Scaling ϵ yields the result.

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Theorem: If we set $r_e = p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}$,The electrical flow oracle satisfies $\sum_{e\in E} p_e^t f(e) \leq (1+\epsilon)\sum_{e\in E} p_e^t$ and in addition, the width is $\rho = O(\sqrt{\frac{m}{\epsilon}})$

$$\sum_{e\in E} p_e^t f(e)^2 \leq (1+\epsilon)\sum_{e\in E} p_e^t \qquad (2)$$

To prove (2), we note that f is an electrical flow, so it minimizes the energy. So we have:

$$\sum_{e\in E} p_e^t f(e)^2 \leq \sum_{e\in E} r_e f(e)^2 = \sum_{e\in E}\left(p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}\right) f(e)^2 \leq \sum_{e\in E}\left(p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}\right) f^*(e)^2$$

$$\leq \sum_{e\in E}\left(p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}\right) = \sum_{e\in E} p_e^t + \epsilon \sum_{e\in E} \frac{\epsilon \sum_{e\in E} p_e^t}{m} = (1+\epsilon)\sum_{e\in E} p_e^t$$

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Theorem: If we set $r_e = p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m}$,The electrical flow oracle satisfies $\sum_{e\in E} p_e^t f(e) \le (1+\epsilon)\sum_{e\in E} p_e^t$ and in addition, the width is $\rho = O(\sqrt{\frac{m}{\epsilon}})$

To prove $\rho = O(\sqrt{\frac{m}{\epsilon}})$, observe that

$$\rho = \max(f(e) - 1)$$

Recall

$$\sum_{e\in E} r_e f(e)^2 \le (1+\epsilon)\sum_{e\in E} p_e^t \quad \Rightarrow \quad f(e)^2 \le \frac{(1+\epsilon)\sum_{e\in E} p_e^t}{r_e}$$

But $r_e = p_e^t + \frac{\epsilon \sum_{e\in E} p_e^t}{m} \ge \frac{\epsilon \sum_{e\in E} p_e^t}{m} \Rightarrow \frac{\sum_{e\in E} p_e^t}{r_e} \le \frac{m}{\epsilon}$ and so

$$f(e)^2 \le \frac{(1+\epsilon)m}{\epsilon} \Rightarrow f(e) = O(\sqrt{\frac{m}{\epsilon}})$$

$\widetilde{O}(m^{1.5})$ MAX-FLOW ALGORITHM

Theorem: Max flow can be solved in $\widetilde{O}(m^{1.5})$ time.

The Tardos framework takes $O\left(\frac{\rho}{\epsilon^2}\right) = O\left(\frac{\sqrt{m}}{\epsilon^{2.5}}\right)$ **iterations**.

Each iteration requires computing effective resistance on a graph. Can be done in $\tilde{O}(m)$ as discussed earlier.

Binary search on max flow value takes $\tilde{O}(1)$ time.

Overall $\tilde{O}\left(m^{1.5}\right)$ time for constant ϵ.

Same algorithm can be improved to $\tilde{O}\left(m^{4/3}\right) = \tilde{O}\left(m^{1.333..}\right)$ by being smarter on using the inequalities, but analysis is more tedious.

QUESTIONS?