ORF 523
Convex and Conic Optimization

Amir Ali Ahmadi
Princeton, ORFE

Lecture 1

What is this course about?

- The mathematical and algorithmic theory of making optimal decisions subject to constraints.

- **Common theme of every optimization problem:**
  - You make decisions and choose one of many alternatives.
  - You hope to maximize or minimize something (you have an objective).
  - You cannot make arbitrary decisions. Life puts constraints on you.

- This pretty much encompasses everything that you do when you are awake. But let's see a few concrete examples…

Examples of optimization problems

In finance

- In what proportions to invest in 500 stocks?
  - To maximize return.
  - To minimize risk.
  - No more than 1/5 of your money in any one stock.
  - Transactions costs < $70.
  - Return rate > 2%.

In control engineering

- How to drive an autonomous vehicle from A to B?
  - To minimize fuel consumption.
  - To minimize travel time.
  - Distance to closest obstacle > 2 meters.
  - Speed < 40 miles/hr.
  - Path needs to be smooth (no sudden changes in direction).

Examples of optimization problems

In economics

- How to play a strategic game?
  - To maximize payoff.
  - To maximize social welfare.
  - Be at a (Nash) equilibrium.
  - Randomize between no more than five strategies.

In machine learning

- How to assign likelihoods to emails being spam?
  - To minimize probability of a false positive.
  - To penalize overfitting on training set.
  - Probability of false negative < .15.
  - Misclassification error on training set < 5%.

- So the question is not
  - Which problems are optimization problems? (The answer would be everything.)
- A much better question is
  - *Which optimization problems can we solve?*

- This is what this course is about.
- We will formalize what we mean by “solve”.
- We’ll see some of the most successful modern optimization tools available to solve a broad class of problems.
- We will also see problems that we simply cannot solve.
  - Nevertheless, we’ll introduce strategies for dealing with them.
- There will be a number of applications…

Prerequisites

- Linear optimization (e.g., at the level of ORF 522)
  - Familiarity with modeling, linear programming, and basic concepts of optimization.
- Linear algebra
- Multivariate calculus
- Familiarity with MATLAB or similar software (e.g., Python)
  - Easy to pick up

Tentative list of topics

- Optimality conditions in nonlinear programming
- Convex analysis (a good dose)
- Duality and infeasibility certificates
- Computational complexity
  - Focus on complexity in numerical optimization
- Conic programming
- More in depth coverage of semidefinite programming
- A module on combinatorial optimization
- Selected topics:
  - Robust optimization
  - Polynomial optimization
  - Sum of squares programming
  - Optimization in dynamical systems
  - Conic-optimization based approximation algorithms

Agenda for today

- Meet your teaching staff & classmates
- Get your hands dirty with algorithms
  - Game 1
  - Game 2
- Course logistics and expectations

Meet your teaching staff

- **Amir Ali Ahmadi** (Amir Ali, or Amirali, is my first name)

http://aaa.princeton.edu/ **aaa@p...**

- I am a Professor at ORFE, and affiliated faculty at COS, ECE, MAE, PACM, CSML
- I came to Princeton from MIT, EECS, after a fellowship at IBM Research
- I was on leave for a while, but now back to teaching!

- **Abraar Chaudhry (1/2 AI)**
- ORFE final-year grad student
- azc@p...

- **Yixuan Hua (Full AI)**
- ORFE second-year grad student
- yh7422@p...

Office hours:

- **Yixuan:** Mon 5-7pm, Sherrerd 122
- **AAA & Abraar:** Wed 3-6pm, Sherrerd 122
- Please also use Ed Discussion via Canvas

Meet your classmates!

- Your name?
- Department?
- Year?
- Maybe a bit of background?

Let’s get to the games!

Meet your fellow Princetonians!

- The green check marks tell you when your visitors are available.
- You want to meet as many of them as you can, for 15 minutes each.
- 20 visitors, 20 time slots. How many can you meet?

- Let me start things off for you. Here is 15 meetings:

- Can you do better? How much better?
- You all get a copy of this Doodle on the handout. You have 5 minutes!

You tell me, I draw…

A good attempt

- 18 meetings!

- Can you do better?

An even better attempt

- 19 meetings!

- Can you do better?
- How would you convince someone that it's impossible to do better?

19 is the best possible!

- Proof by magic:

- Do you see what's happening?

19 is the best possible!

- There are 19 red arrows.
- Each green checkmark "touches" at least one of them (by going either up or left).
- If you could choose 20 green checkmarks, at least two of them would have to touch the same arrow.
- And here is the magic: such a proof is *always* possible!

A related problem: shipping oil!

Before we get to our second game, let's look at another problem which may look more familiar to you.

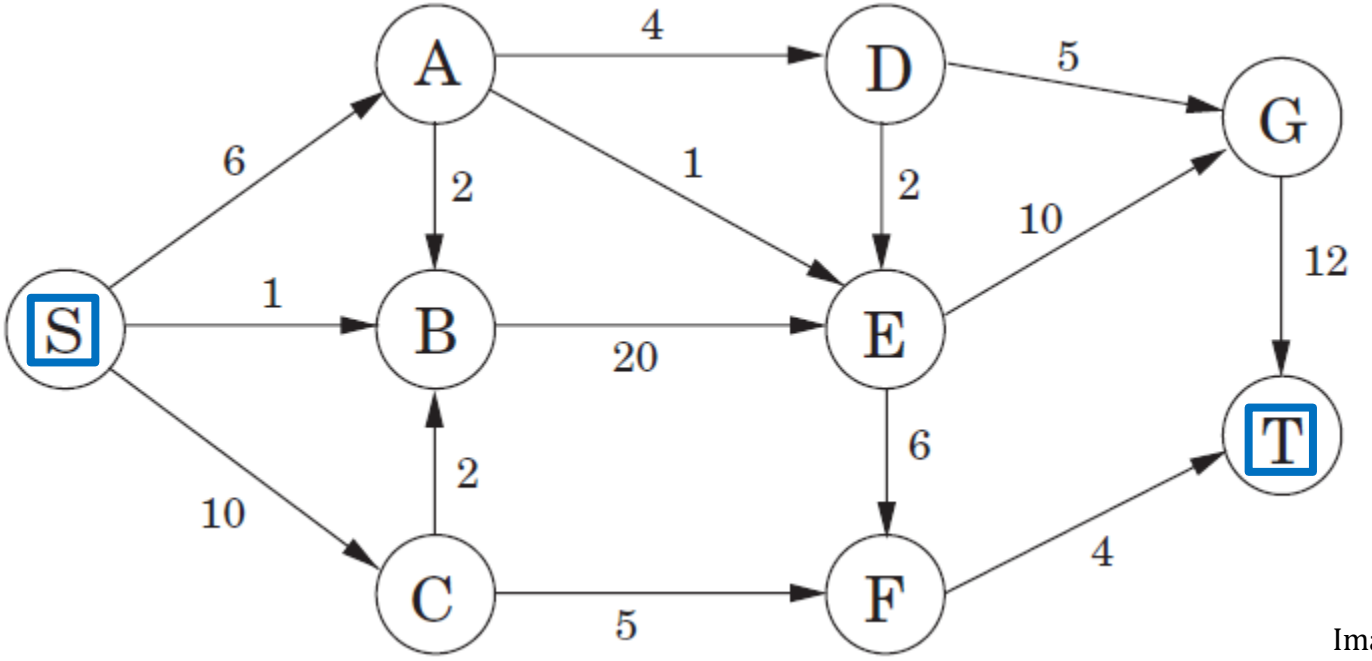
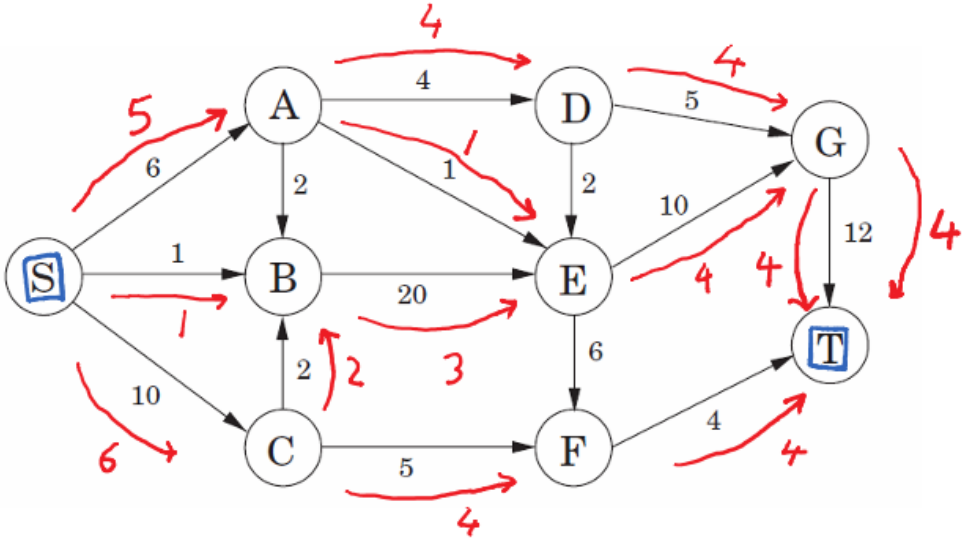


Image credit: [DPV08]

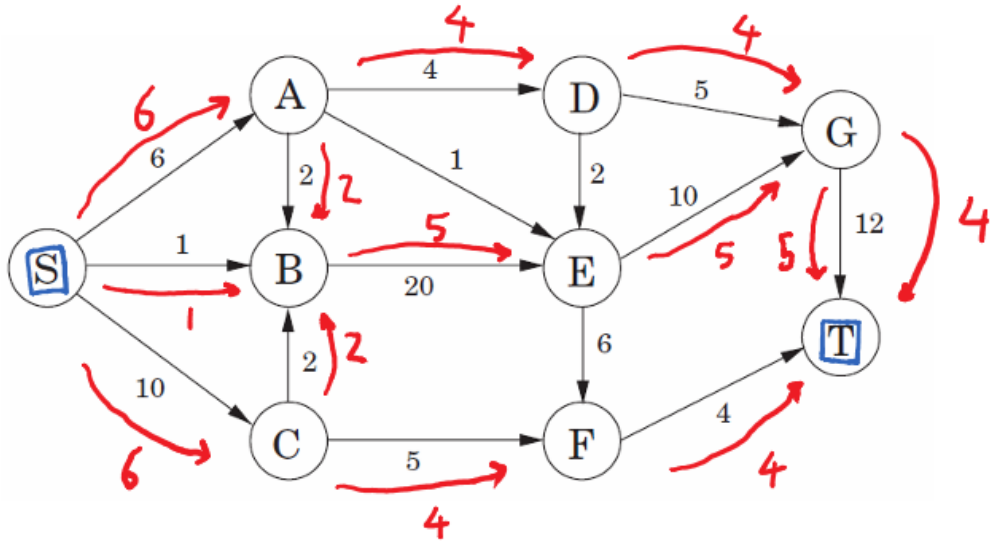
- **Rules of the problem:**
  - Cannot exceed capacity on the edges.
  - For each node, except for S ant T, flow in = flow out (i.e., no storage).
  - **Goal:** ship as much oil as you can from S to T.

A couple of good attempts



$4+4+4=12$

- Can you do better?

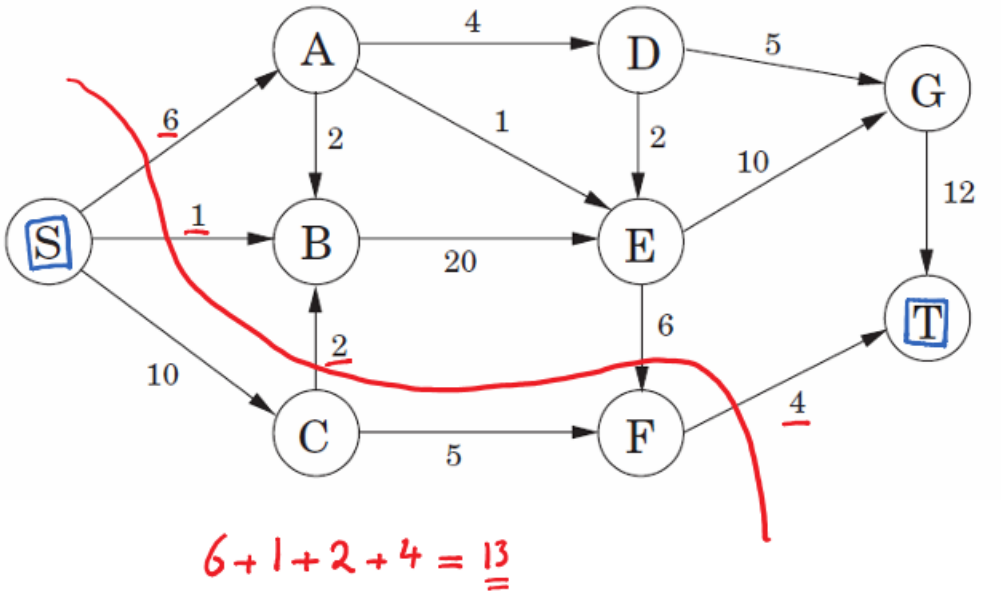


$5+4+4=13$

- Can you do better?
- How can you convince someone that it's impossible to do better?

13 is the best possible!

- Proof by magic:

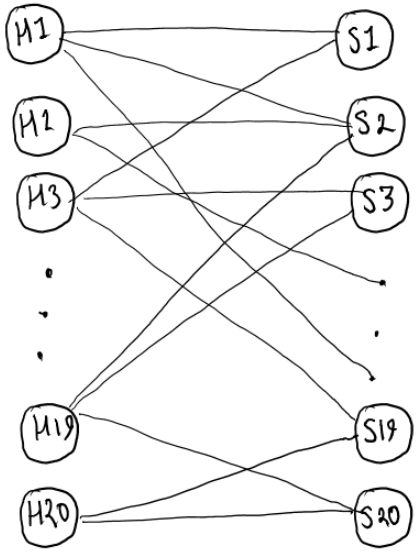


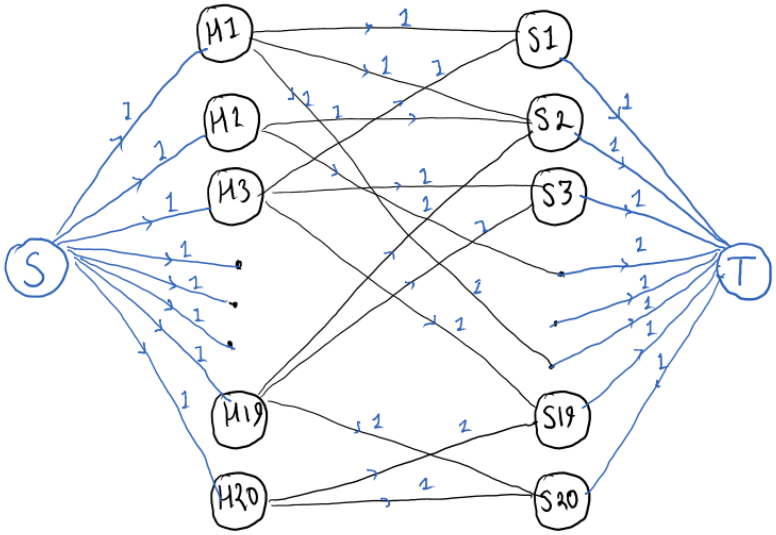
- The rabbit is the red "cut"!
- Any flow from S to T must cross the red curve.
- So it can have value at most 13.
  - And here is the magic: such a proof is *always* possible!

- What does any of this have to do with the Doodle problem?

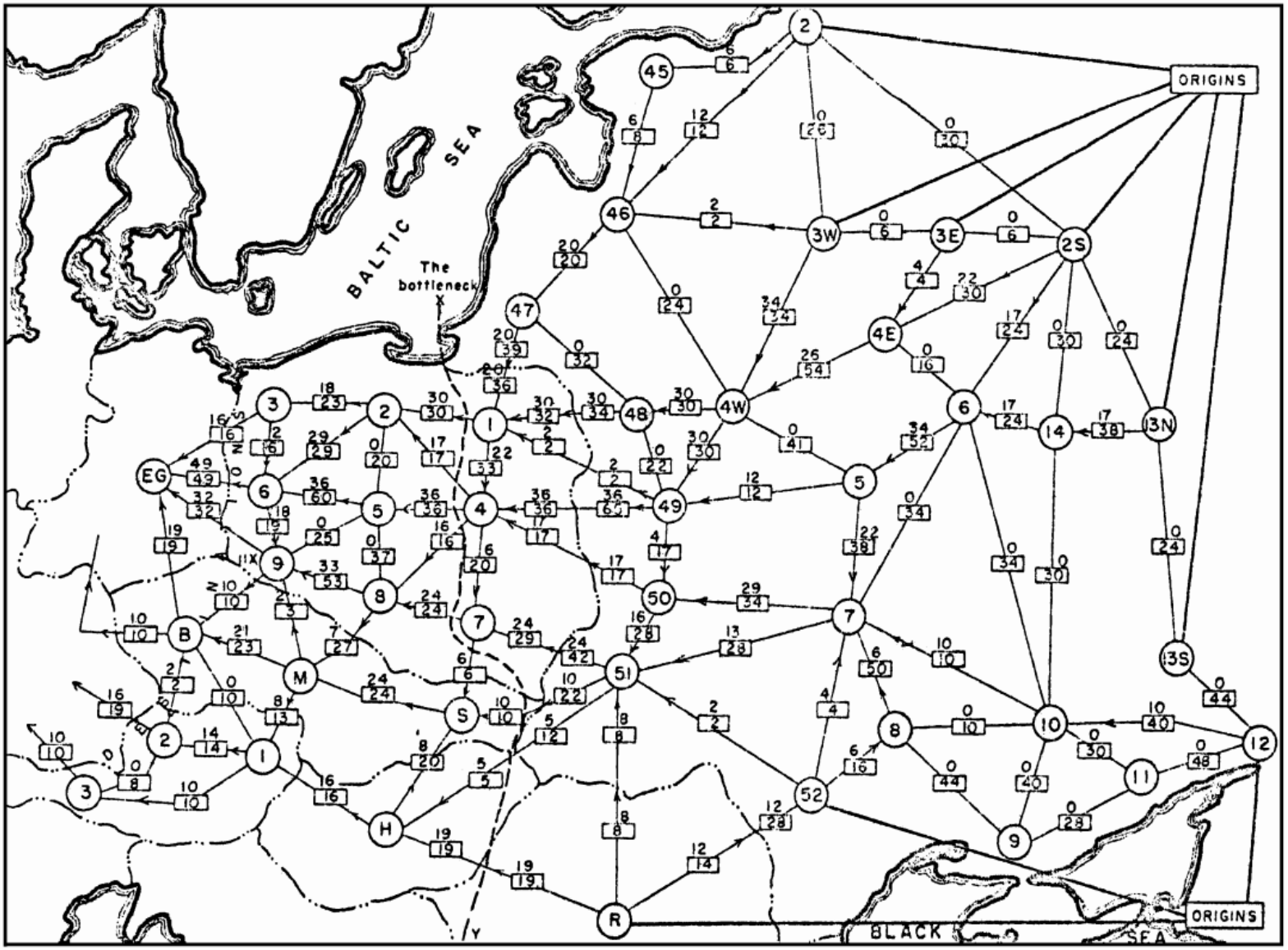
From Doodle to Max-flow

- The idea of **reductions**
- They'll come up often

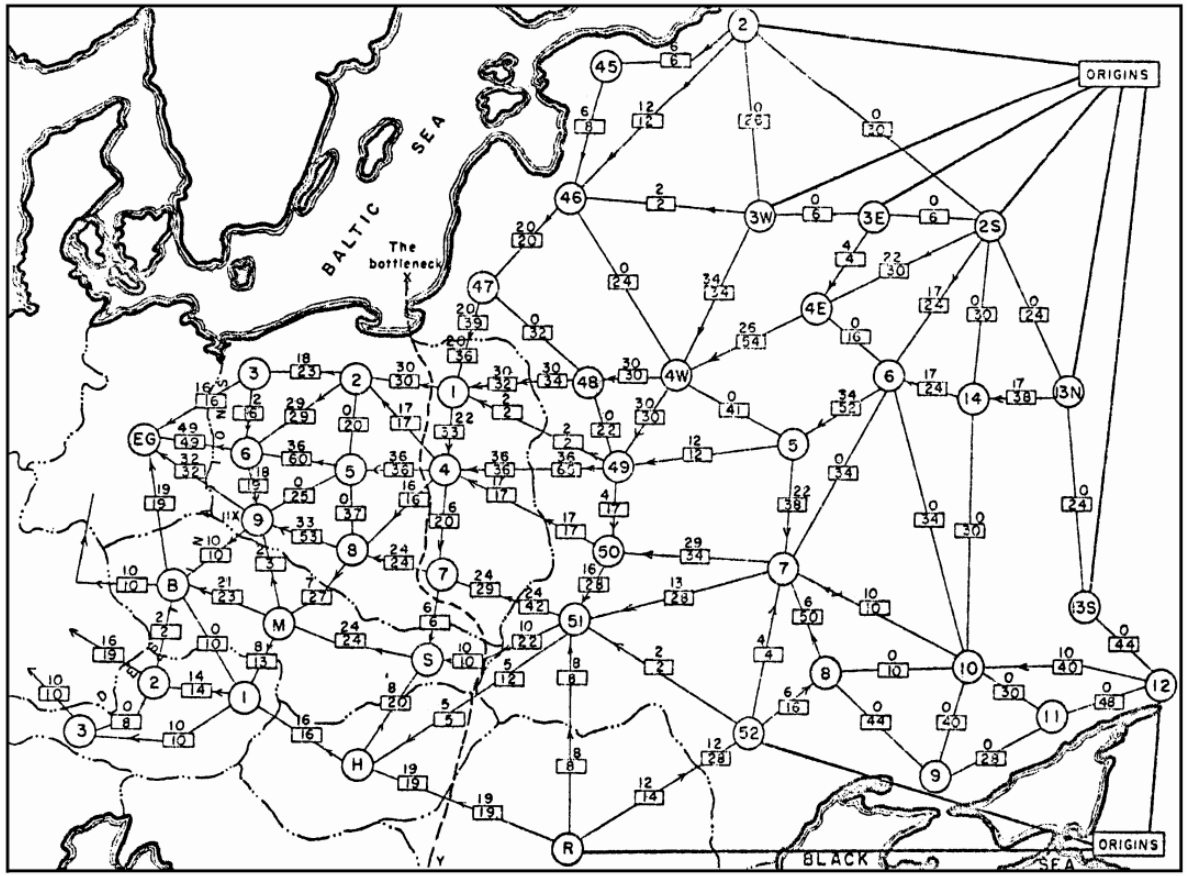




A real-world instance of max-flow

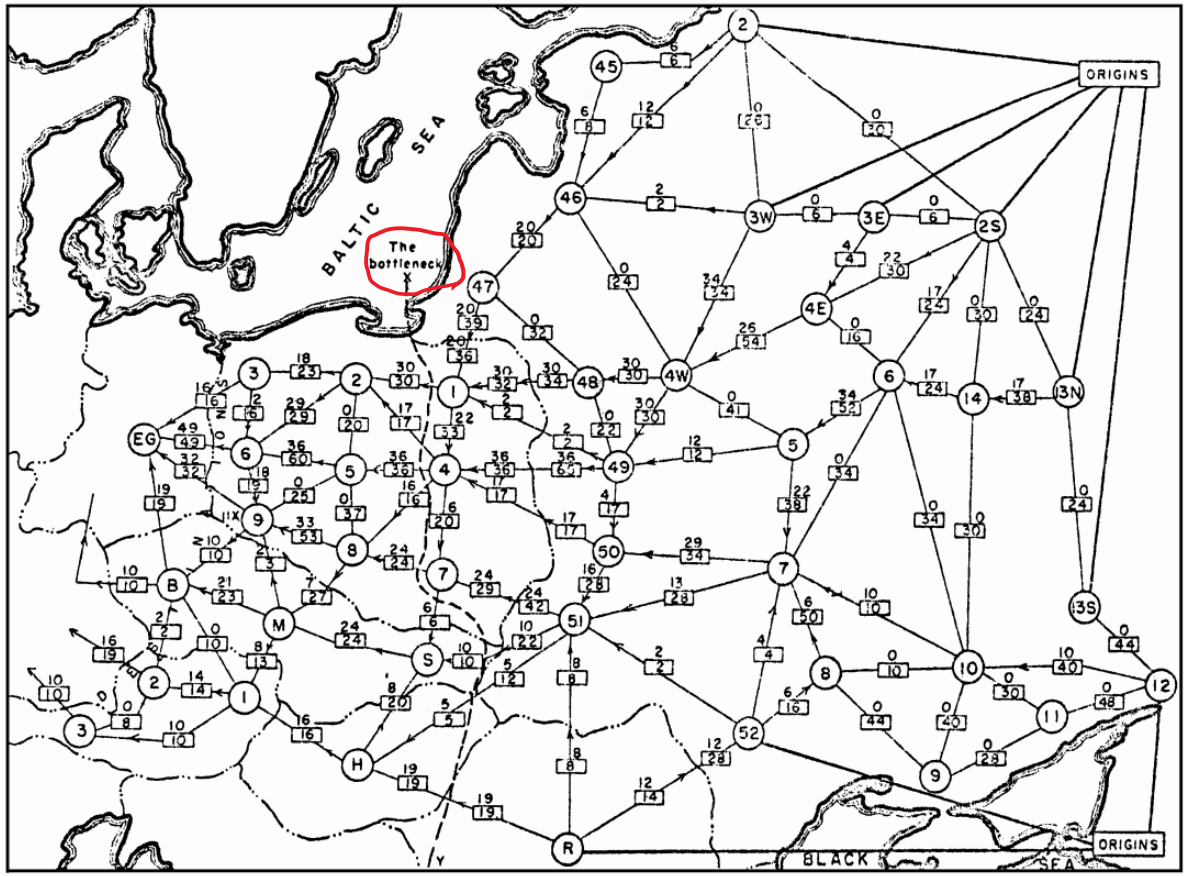


- How long do you think an optimization solver would take (on my laptop) to find the best solution here?
- How many lines of code do you think you have to write for it?
- How would someone who hasn't seen optimization approach this?
  - Trial and error?
  - Push a little flow here, a little there…
  - Do you think they are likely to find the best solution?
  - How would they certify it?



A bit of history behind this map

- From a secret report by Harris and Ross (1955) written for the Air Force.
- Railway network of the Western Soviet Union going to Eastern Europe.
- Declassified in 1999.
- Look at the min-cut on the map (called the "bottleneck")!
- There are 44 vertices, 105 edges, and the max flow is 163K.



- Harris and Ross gave a heuristic which happened to solve the problem optimally in this case.
- Later that year (1955), the famous Ford-Fulkerson algorithm came out of the RAND corporation. The algorithm always finds the best solution (for rational edge costs).

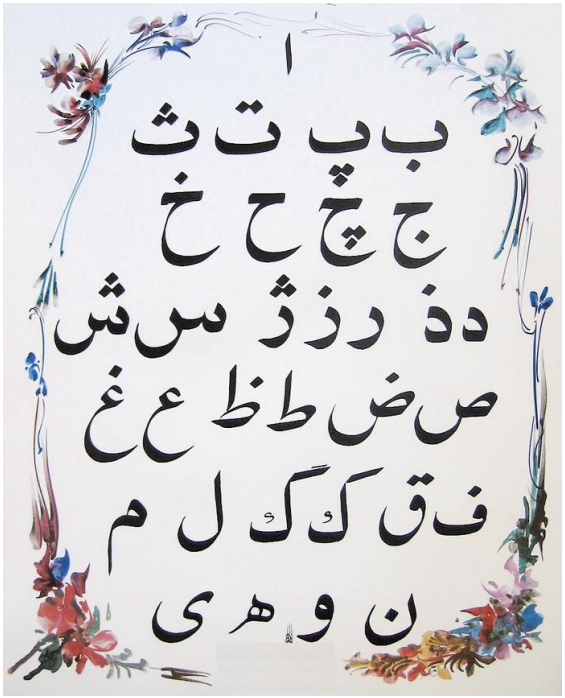
More on this history: [Sch05]

Let’s look at our second problem

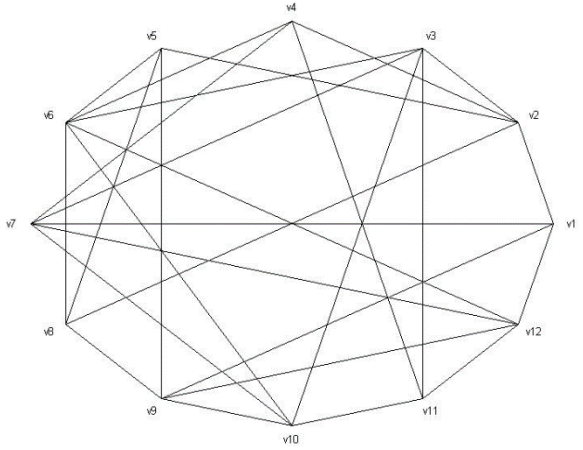
…and tell me which one you thought was easier

Robust-to-noise communication

- You are given a set of letters from an alphabet.
- Want to use them for communication over a noisy channel.
- Some letters look similar and can be confused at the receiving end because of noise. (Notion of similarity can be formalized; e.g., think of Hamming distance.)

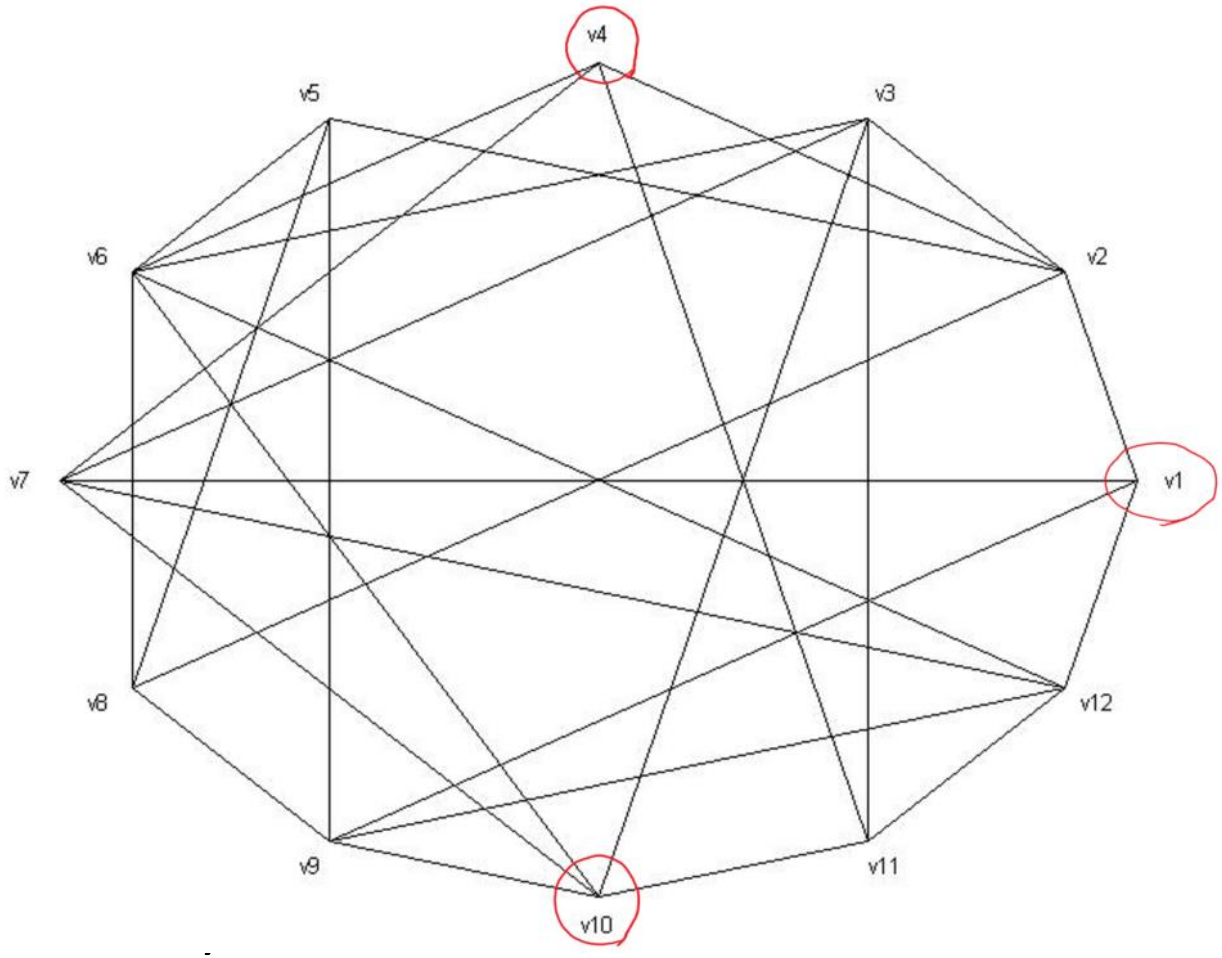


- Let's draw a graph whose nodes are our letters. There is an edge between two nodes if and only if the letters can be confused.
- We want to pick the maximum number of letters that we can safely use for communication (i.e., no two should be prone to confusion).
  - What are we looking for in this graph?



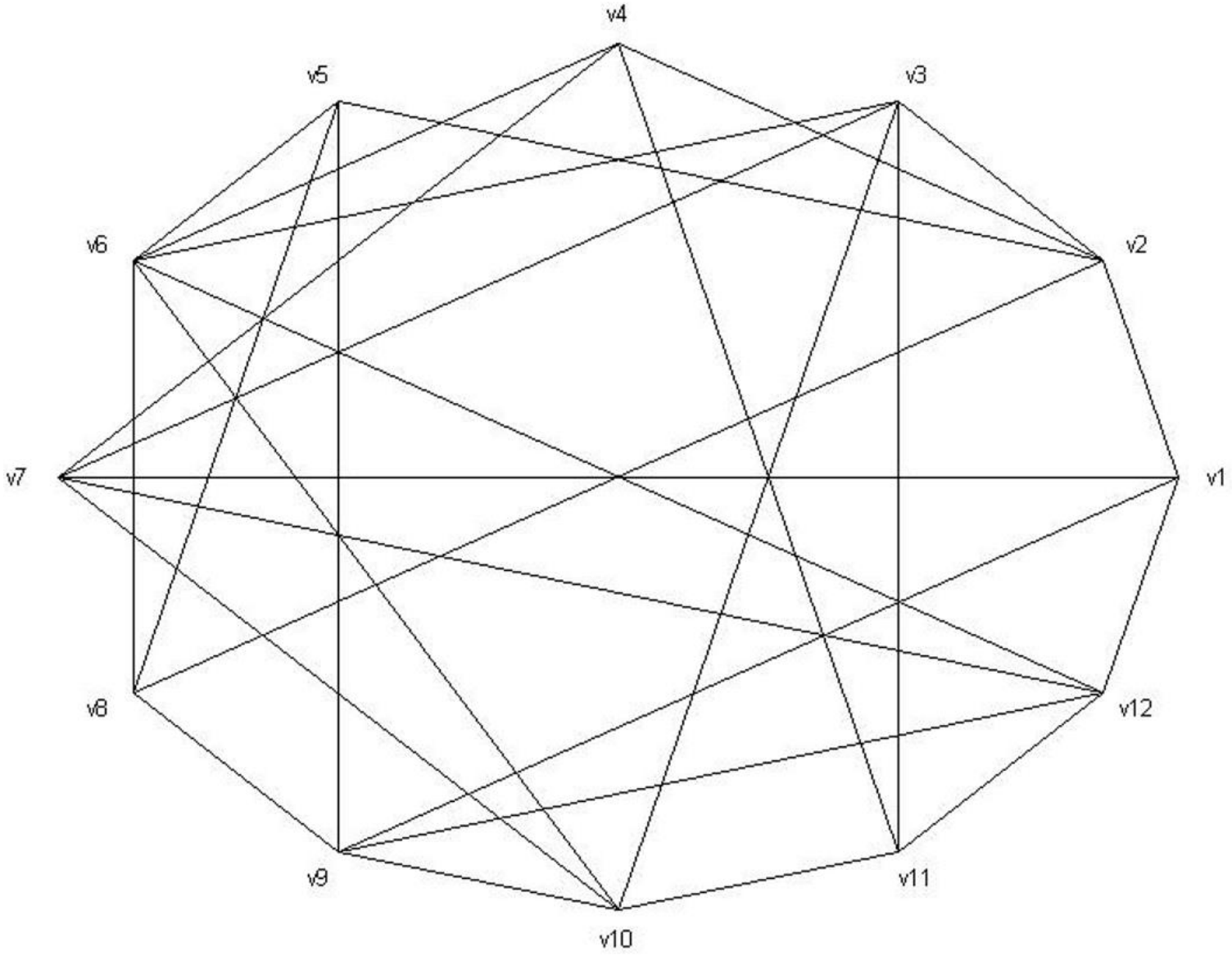
- The largest "stable set" (aka "independent set")!

- Let me start things off for you. Here is a stable set of size 3:

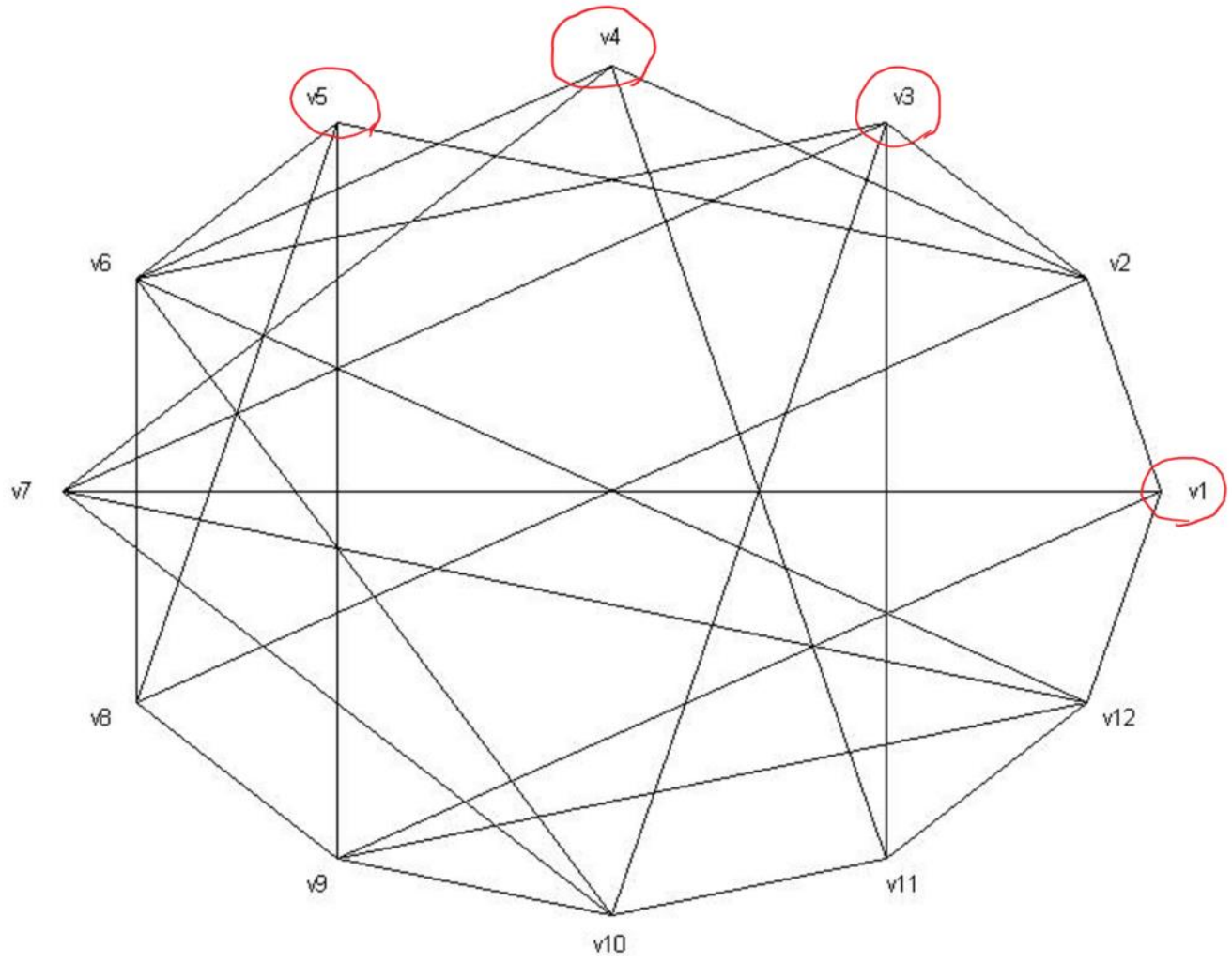


- You all get a copy of this graph on the handout.
  - You have 5 minutes!

You tell me, I draw…



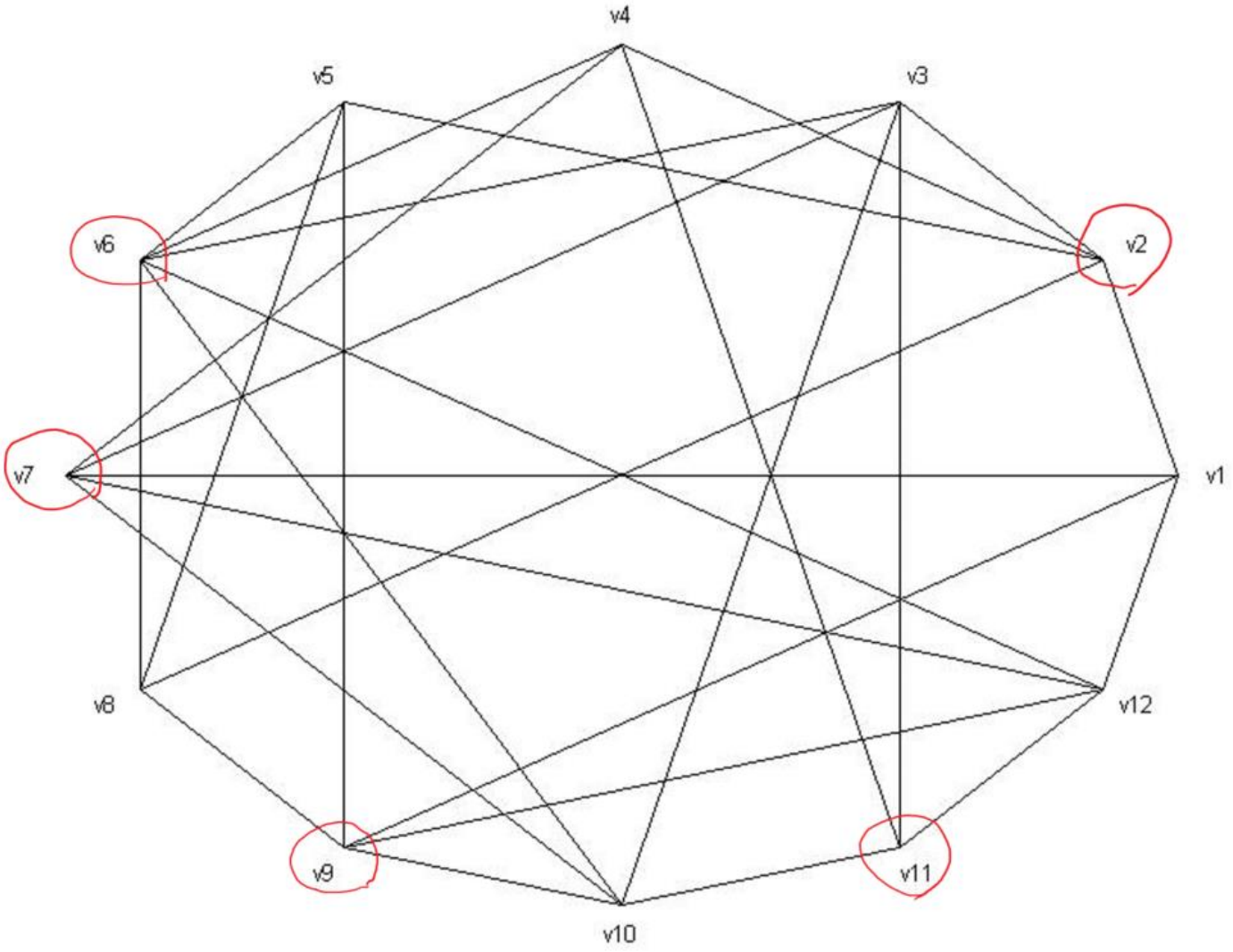
A couple of good attempts



Size 4

- Can you do better?

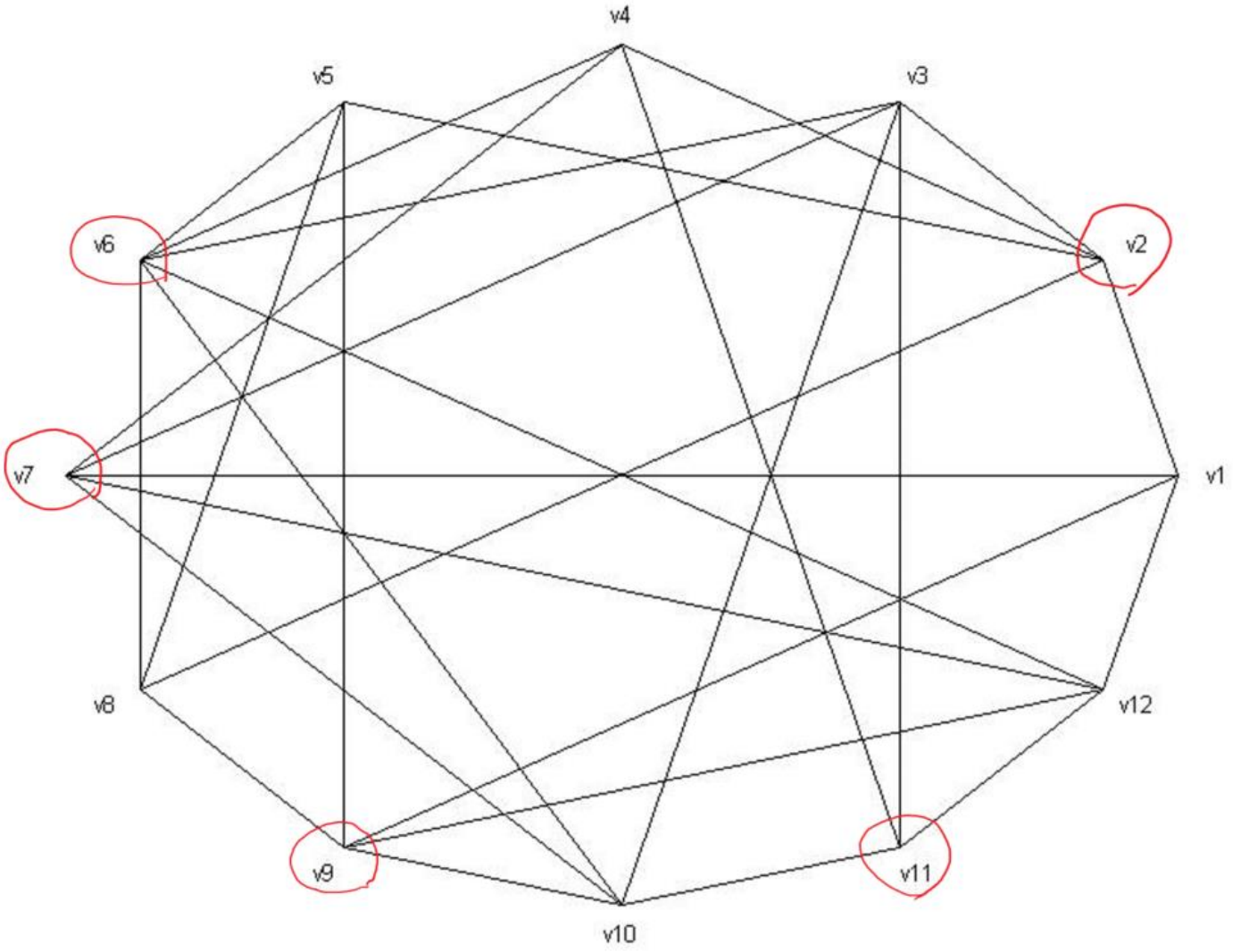
A couple of good attempts



Size 5

- Can you do better?

A couple of good attempts



Size 5

- Tired of trying?
- Is this the best possible?

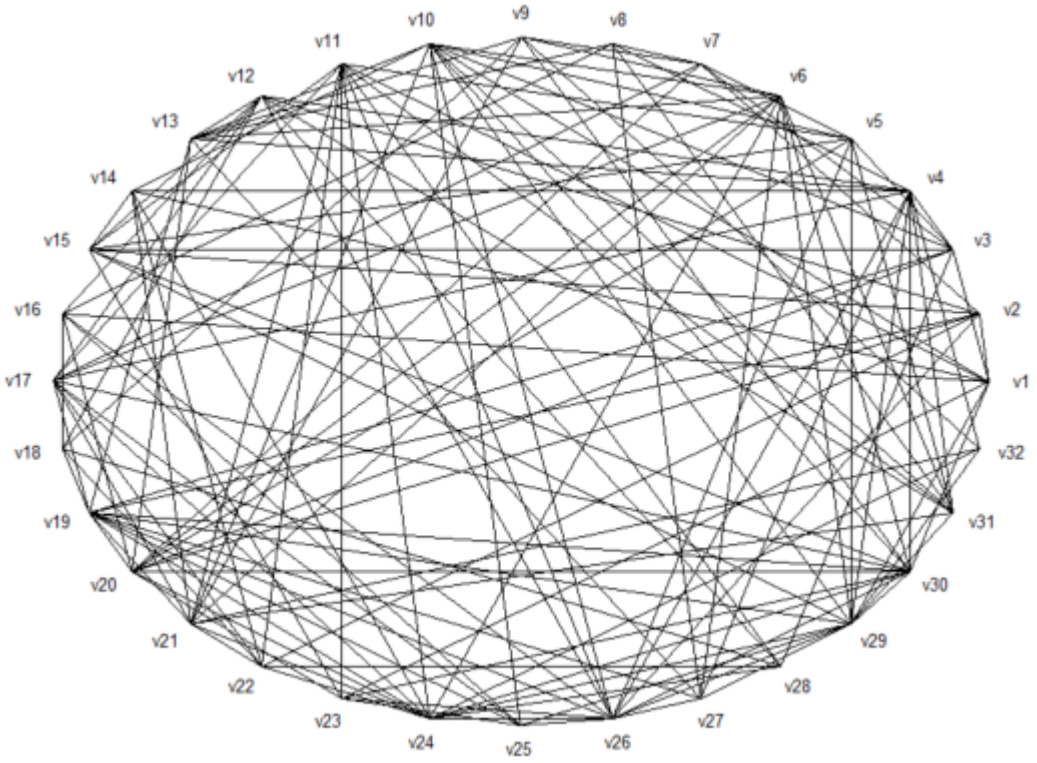
5 is the best possible!

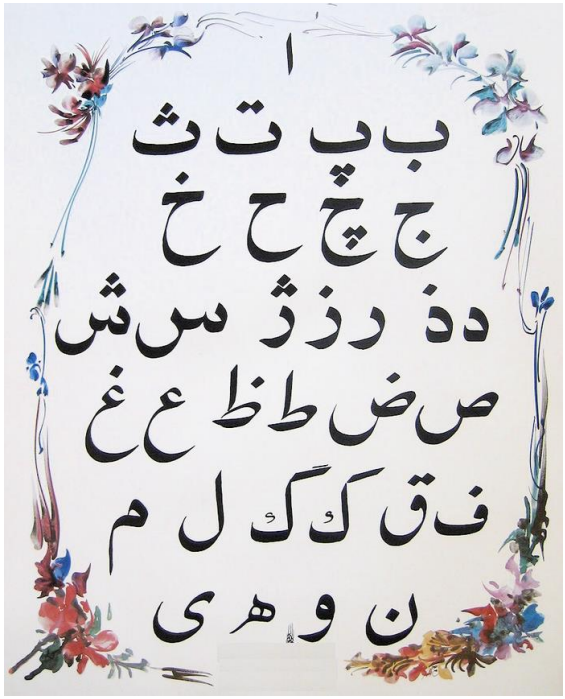
- Proof by magic?

  - Unfortunately not ☹
  - No magician in the world has pulled out such a rabbit to this day! (By this we mean a trick that would work on *all* graphs.)

- Of course, there is always a proof:
  - Try all possible subsets of 6 nodes.
  - There are 924 of them.
  - Observe that none of them work.
- But this is no magic. It impresses nobody. We want a "short" proof. (We will formalize what this means.) Like the one in our Doodle/max-flow examples.
- Let's appreciate this further…

What our graph can look like with 32 letters

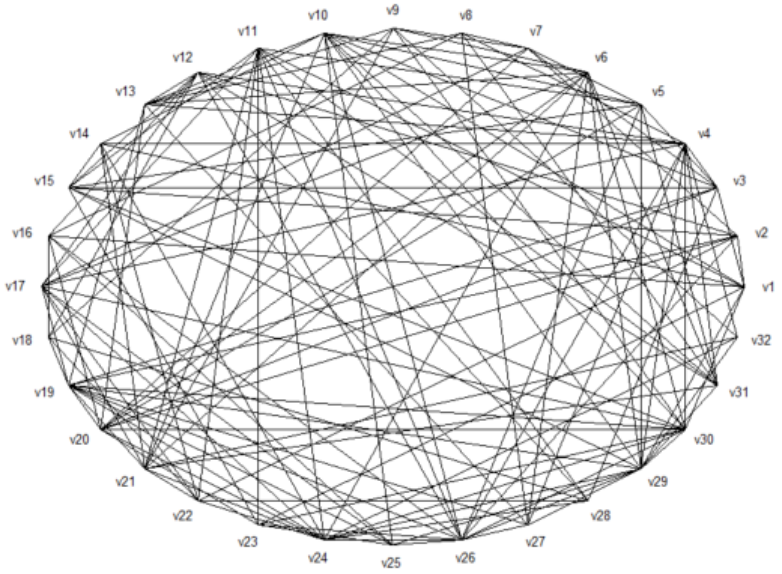




- Maximum stable set anyone? ;)
- Is there a stable set of size 16?
- Want to try all possibilities? There are **over 600 million** of them!!
- If the graph had 100 nodes, there would be **over 10^{18} possibilities** to try!

But there is some good news

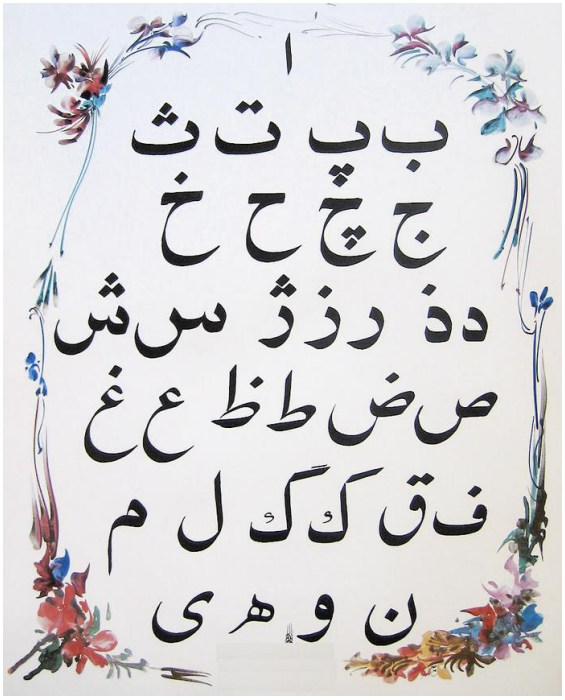
- Even though finding the best solution always may be too much to hope for, techniques from optimization (and in particular from the area of *convex optimization*) often allow us to find high-quality solutions with performance guarantees.

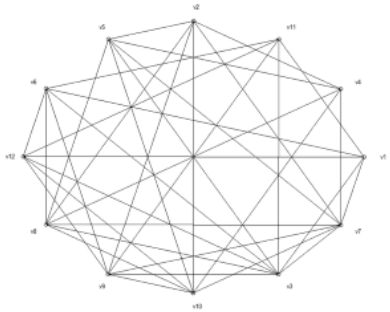


- For example, an optimization algorithm may quickly find a stable set of size 15 for you.
- You really want to know if 16 is impossible. Instead, another optimization algorithm (or sometimes the same one) tells you that 18 is impossible.
- This is very useful information! You know you got 15, and no one can do better than 18.
  - **We sill see a lot of convex optimization in this class!**

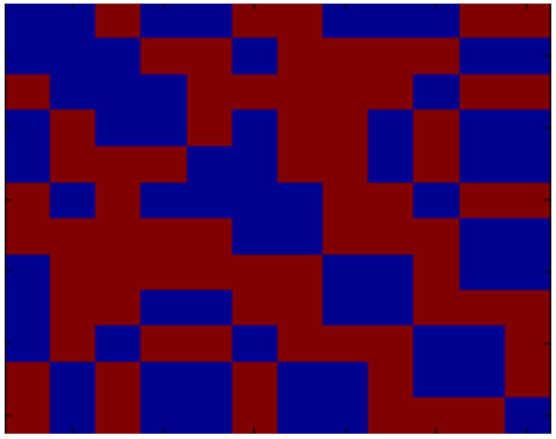
A related problem: capacity of a graph

- Suppose instead of single letters, we wanted to send k-tuples of letters. How many k-tuples can we collect such that no two among them can be confused?
- Two k-tuple can be confused if for each $1 \leq i \leq k$, their i-th letters can be confused (or are equal).
- The answer is the stability number of the "k-th power of the graph", in the sense of *strong graph products* (k-th Kronecker power of the adjacency matrix).
- Example: k=2.

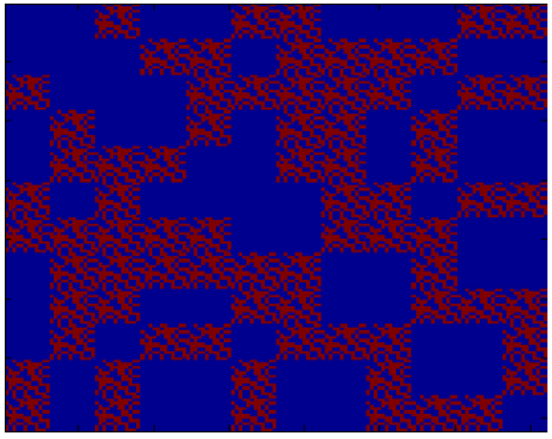




G

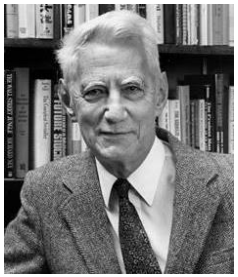


A_G



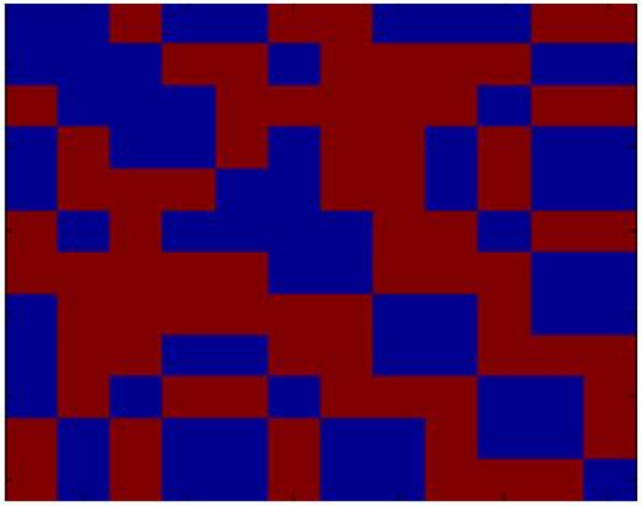
$A_{G^2} = A_G \otimes A_G$

Capacity of a graph

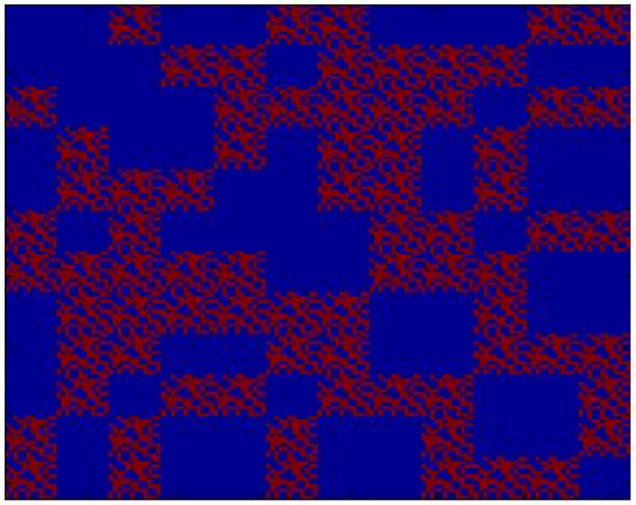


$$\Theta(G) = \lim_{k \to \infty} \sqrt[k]{\alpha(G^k)}$$

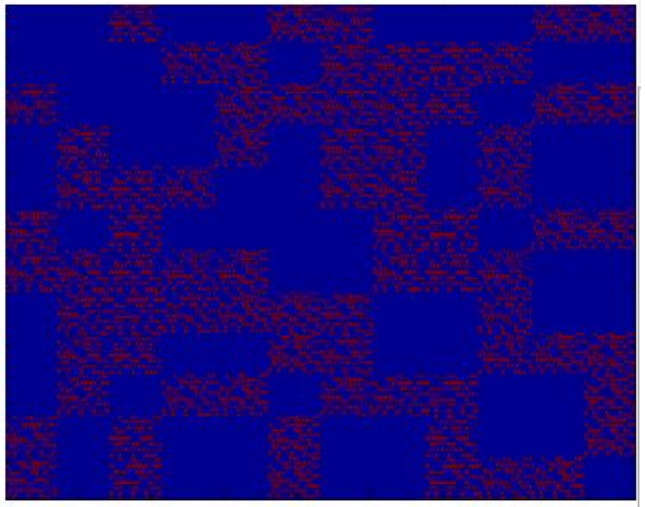
(α: size of max stable set)



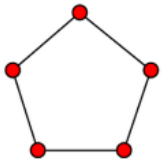
G



G^2



G^3



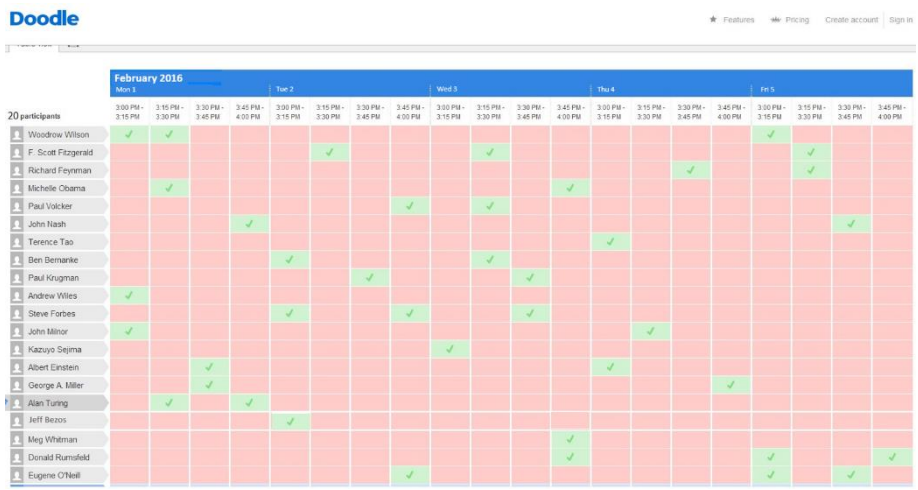
$\Theta(C_5) = \sqrt{5}$

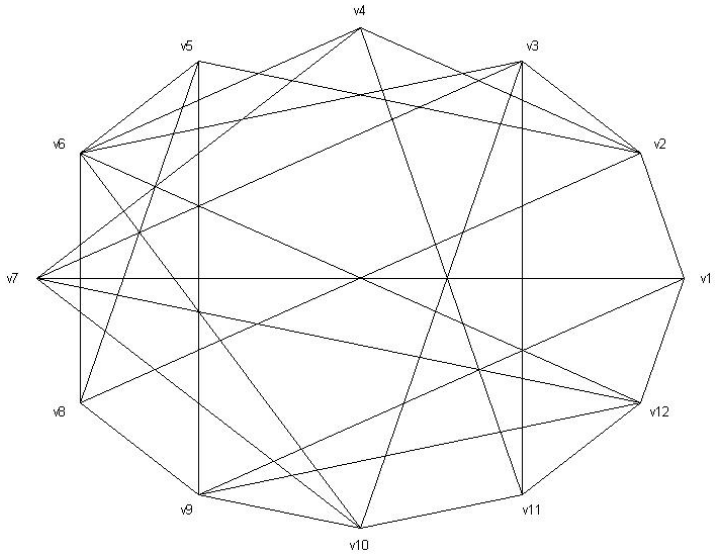
$\Theta(C_7) = ?$ (open)

using semidefinite programming

(Automatic "A" in ORF 523)

Which of the two problems was harder for you?



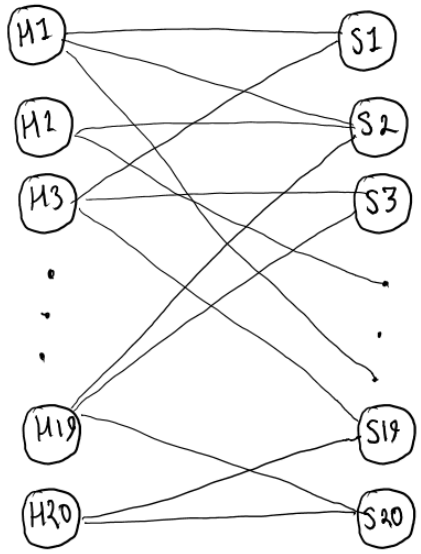


- Not always obvious. A lot of research in optimization and computer science goes into distinguishing the "tractable" problems from the "intractable" ones.
- The two brain teasers actually just gave you a taste of the P vs. NP problem. (If you haven't seen these concepts formally, that's OK. You will soon.)
- The first problem we can solve efficiently (in "polynomial time").
- The second problem: no one knows. If you do, you literally get $1M!
  - More importantly, your algorithm immediately translates to an efficient algorithm for thousands of other problems no one knows how to solve.

Modelling problems as a mathematical program

Let's revisit our first game

- What were your decision variables?
- What were your constraints?
- What was your objective function?



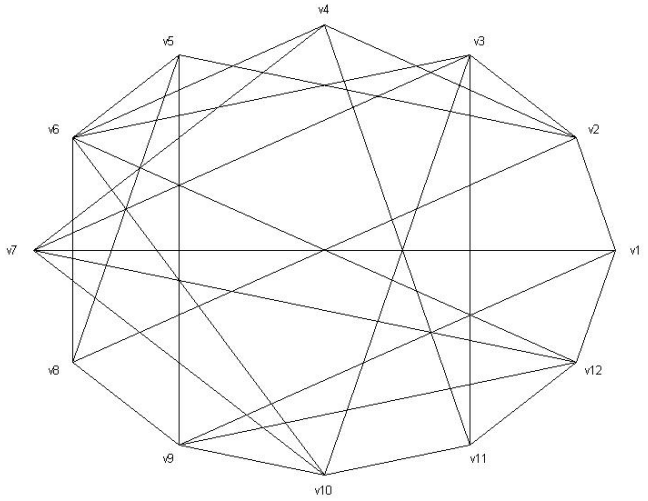
Variables: x_e (one per edge)

$G(V,E)$ $\quad \max \sum_{e \in E} x_e$

$\sum_{e \ni v} x_e \leq 1 \quad \forall v \in V$

$x_e(1-x_e)=0 \quad \forall e \in E$

Let's revisit our second game



- What were your decision variables?
- What were your constraints?
- What was your objective function?

variables: x_i (one per node)

$G(V,E)$ $\quad \max \sum x_i$

$x_i + x_j \leq 1 \quad$ if $(i,j) \in E$

$x_i(1-x_i) = 0 \quad \forall i \in V$

Why one hard and one easy? How can you tell?

Variables: x_e (one per edge)

$G(V,E)$

$$\max \sum_{e \in E} x_e$$

$$\sum_{e \ni v} x_e \leq 1 \quad \forall v \in V$$

$$x_e(1-x_e)=0 \quad \forall e \in E$$

Variables: x_i (one per node)

$G(V,E)$

$$\max \sum x_i$$

$$x_i + x_j \leq 1 \quad \text{if } (i,j) \in E$$

$$x_i(1-x_i)=0 \quad \forall i \in V$$

- **Caution:** just because we can write something as a mathematical program, it doesn't mean we can solve it.

Fermat's Last Theorem

- Can you give me three positive integers x, y, z such that

$$x^2 + y^2 = z^2?$$

- Sure: (3, 4, 5) (5, 12, 13) (8, 15, 17) (7, 24, 25)
  (20, 21, 29) (12, 35, 37) (9, 40, 41) (28, 45, 53)

And there are infinitely many more…

- How about $x^3 + y^3 = z^3?$
- How about $x^4 + y^4 = z^4?$
- How about $x^5 + y^5 = z^5?$

Fermat's Last Theorem

- Fermat's conjecture (1637):

For $n \geq 3$, the equation $x^n + y^n = z^n$ has no solution over positive integers.

- Proved in 1994 (357 years later!) by Andrew Wiles.

(Was on the faculty in our math department until a few years ago.)

Arithmeticorum Liber II. 61

interuallum numerorum 2. minor autem 1 N. atque ideo maior 1 N. + 2. Oportet itaque 4 N. + 4. triplos esse ad 2. & adhuc superaddere 10. Ter igitur 2. adsciti vnitatibus 10. æquatur 4 N. + 4. & fit 1 N. 3. Erit ergo minor 3. maior 5. & satisfaciunt quæstioni.

IN QVAESTIONEM VII.

CONDITIONIS appositæ eadem ratio est quæ & appositæ præcedenti quæstioni, nil enim aliud requirit quàm vt quadratus interualli numerorum sit minor interuallo quadratorum, & Canones iidem hic etiam locum habebunt, vt manifestum est.

QVÆSTIO VIII.

PROPOSITVM quadratum diuidere in duos quadratos. Imperatum sit vt 16. diuidatur in duos quadratos. Ponatur primus 1 Q. Oportet igitur 16 − 1 Q. æquales esse quadrato. Fingo quadratum à numeris quotquot libuerit, cum defectu tot vnitatum quod continet latus ipsius 16. esto à 2 N. − 4. ipse igitur quadratus erit 4 Q. + 16. − 16 N. hæc æquabuntur vnitatibus 16 − 1 Q. Communis adiiciatur vtrimque defectus, & à similibus auferantur similia, fient 5 Q. æquales 16 N. & fit 1 N. $\frac{16}{5}$. Erit igitur alter quadratorum $\frac{256}{25}$. alter verò $\frac{144}{25}$ & vtriusque summa est $\frac{400}{25}$ seu 16. & vterque quadratus est.

OBSERVATIO DOMINI PETRI DE FERMAT.

Verum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos & generaliter nullam in infinitum vltra quadratum potestatem in duos eiusdem nominis fas est diuidere cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.

QVÆSTIO IX.

RVRSVS oporteat quadratum 16 diuidere in duos quadratos. Ponatur rursus primi latus 1 N. alterius verò quotcunque numerorum cum defectu tot vnitatum, quot constat latus diuidendi. Esto itaque 2 N. − 4. erunt quadrati, hic quidem 1 Q. ille verò 4 Q. + 16. − 16 N. Cæterum volo vtrumque simul æquari vnitatibus 16. Igitur 5 Q. + 16. − 16 N. æquatur vnitatibus 16. & fit 1 N. $\frac{4}{5}$ erit

H iii

Fermat's Last Theorem

- Fermat's conjecture (1637):

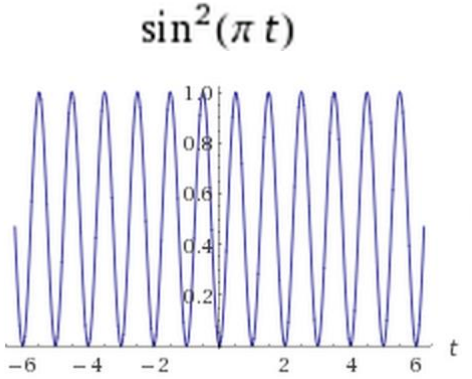
For $n \geq 3$, the equation $x^n + y^n = z^n$ has no solution over positive integers.

- Consider the following optimization problem (mathematical program):

$$\min_{x,y,z,n} \quad (x^n + y^n - z^n)^2$$

$$\text{s.t.} \quad x \geq 1,\ y \geq 1,\ z \geq 1,\ n \geq 3,$$

$$\sin^2 \pi n + \sin^2 \pi x + \sin^2 \pi y + \sin^2 \pi z = 0.$$



- Innocent-looking optimization problem: 4 variables, 5 constraints.
- If you could show the optimal value is non-zero, you would prove Fermat's conjecture!

Course objectives

- The skills I hope you acquire:
- Ability to view your own field through the lens of optimization and computation
  - To help you, we'll draw basic applications from operations research, dynamical systems, finance, machine learning, engineering, …
- Comfort with proofs in convex analysis.
- Improved coding abilities (in e.g. MATLAB, Python, CVX/CVXPY, YALMIP)
  - There will be a computational component on every homework
- Ability to recognize hard and easy optimization problems.
- Ability to rigorously show an optimization problem is hard.
- Solid understanding of conic optimization, in particular semidefinite programming.
- Familiarity with selected topics: robust optimization, polynomial optimization, optimization in dynamical systems, etc.

Software you need to download

- Right away:

  MATLAB (available from the Princeton OIT website)

  You are free to use any other programming language instead (e.g., Python). The solutions that we provide will be in MATLAB (though, I am asking the AIs to add Python solutions).

- In the next couple of weeks (will likely appear on HW#2):

  CVX http://cvxr.com/cvx/

  (If you are comfortable with Python, you are free to use CVXPY instead: https://www.cvxpy.org/)

- Towards the end of the course (for sum of squares optimization):

  YALMIP (MATLAB-based) https://yalmip.github.io/

  (There are alternatives in other languages such as Python, but not as well tested: https://sums-of-squres.github.io/sos)

Course logistics

- Course website: aaa.princeton.edu/orf523

- Your grade:
  - 50% homework (5 or 6 total – biweekly, can drop your lowest score, no extensions allowed)
    - Collaboration policy: you can and are encouraged. Turn in individual psets. Write the name of your collaborators.
  - 20 % Midterm exam (in class – for duration of lecture, a single double-sided page of cheat sheet allowed)
  - 30% Final exam/assignment (think of it as a longer, cumulative homework that needs to be done with no collaboration). Or solve one of our open problems instead.

- Textbooks
  - What matters primarily is class notes. You are expected to take good notes. (I teach on the blackboard (now aka iPad) most of the time.) Georgina Hall (former TA) has provided lecture outlines which are posted on the website.
  - Four references will be posted on the course website if you want to read further – all should be free to download online.

Image credits and references

- [DPV08] S. Dasgupta, C. Papadimitriou, and U. Vazirani. Algorithms. McGraw Hill, 2008.

- [Sch05] A. Schrijver. On the history of combinatorial optimization (till 1960). In “Handbook of Discrete Optimization”, Elsevier, 2005. http://homepages.cwi.nl/~lex/files/histco.pdf