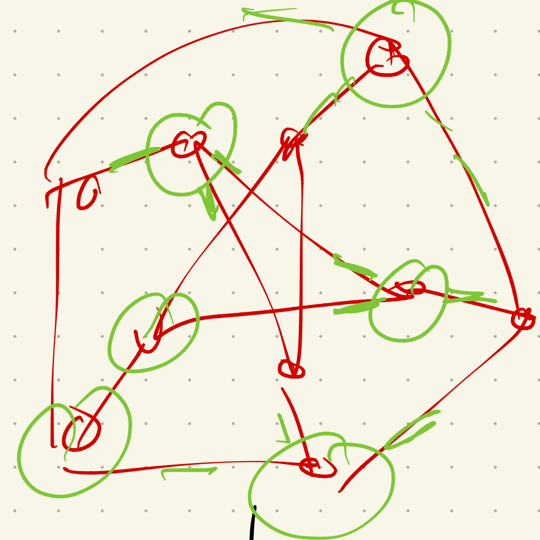
Subset Sum is NP-Hard.



Reduction: Vertex Cover

Input: Graph G & size k

Goal: find k vertices, such that every edge in G is incident to at least one vertex in set

Challenge: Construct a set of numbers, s.t. we can hit a target value

$\iff$ G has ~~indep set~~ Vertex Cover of size k

Recall: base 4

$(32012)_4 = 3\cdot4^4 + 2\cdot4^3 + 0\cdot4^2 + 1\cdot4^1 + 2\cdot4^0$

... 16's 4's 1's

$$\begin{array}{r} 1\,0\,3\,2 \\ +\,2\,3\,1\,2 \\ \hline 1\,0\,0\,1\,0 \end{array}$$

Idea: Use base 4:
force a target T that requires you to use only vertices, but to "cover" edges

Number edges $0 \ldots E-1$, & create a number for subset sum w/ E "digits"

e_0: $b_0 = 0\,0\,\_\,\_\,\_\,\_\,\_\,0\,1$

e_1: $b_1 = 0\,0\,\_\,\_\,\_\,\_\,0\,1\,0$

⋮

e_{E-1}: $b_{E-1} = 0\,1\,0 \ldots 0$

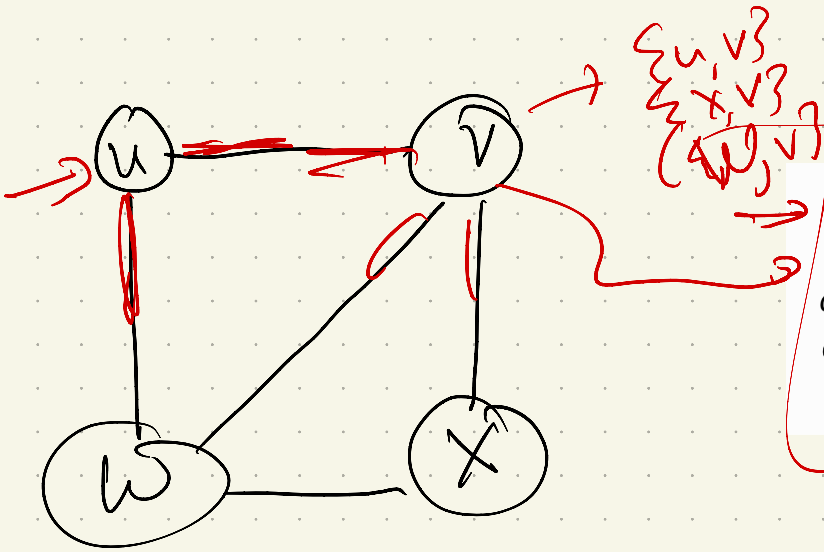
E spots → E+1

For each vertex, make another #:

$a_v := 1[\quad\quad]$ ← $d(v)$ 1's, for its endpoints (all else 0)

E+1 digits

Think of base 4 representation



$a_u := 111000_4 = 1344$		$b_{uv} := 010000_4 = 256$	
$a_v := 110110_4 = 1300$		$b_{uw} := 001000_4 = 64$	
$a_w := 101101_4 = 1105$		$b_{vw} := 000100_4 = 16$	
$a_x := 100011_4 = 1029$		$b_{vx} := 000010_4 = 4$	
		$b_{wx} := 000001_4 = 1$	

→ set of #s,

k of these 4's

$0\,11\cdots1$ (E)

any value > 0

Now, set $T = k \cdot 4^E + \sum_{i=0}^{E-1} 2 \cdot 4^i$

input

2 each

why?

Areas k vertices

Proof: size k VC $\Longrightarrow$ sum to T

$\Rightarrow$ VC: $\exists$ k vertices $v_1, v_2, \ldots, v_k$

s.t. $\forall e \in E$, e is incident to some $v_i \in \{v_1, \ldots, v_k\}$

$\Rightarrow$ (cont)

Pick a subset:

$\Leftarrow$: suppose some subset of #s sums to T. options?

Recall: $T = k \cdot 4^E + \sum_{i=0}^{E-1} 2 \cdot 4^i$

+ a_r = ___

+ b_e = ___

Plus:

Each digit position has only 3 1's across all #s:

So!

Partition:

Given $X = \{x_1, \dots, x_n\}$, can we partition X into A & B (so $A \cup B = X$, $A \cap B = \emptyset$, & $A, B \neq \emptyset$)

s.t.

$$\sum_{x_i \in A} x_i = \sum_{x_j \in B} x_j \quad ?$$

Reduction?

Proof:

Some fun examples

arXiv.org > cs > arXiv:1203.1895

Search or Article ID inside arXiv | All papers | Broaden your search usin

(Help | Advanced search)

Computer Science > Computational Complexity

Classic Nintendo Games are (Computationally) Hard

Greg Aloupis, Erik D. Demaine, Alan Guo, Giovanni Viglietta

(Submitted on 8 Mar 2012 (v1), last revised 8 Feb 2015 (this version, v3))

We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokemon. Our results apply to generalized versions of Super Mario Bros. 1-3, The Lost Levels, and Super Mario World; Donkey Kong Country 1-3; all Legend of Zelda games; all Metroid games; and all Pokemon role-playing games. In addition, we prove PSPACE-completeness of the Donkey Kong Country games and several Legend of Zelda games.

Comments: 36 pages, 36 figures. Fixed some typos. Added NP-hardness results (with proofs and figures) for American SMB2 and Zelda 2

Subjects: **Computational Complexity (cs.CC)**; Computer Science and Game Theory (cs.GT)

Cite as: arXiv:1203.1895 **[cs.CC]**

(or **arXiv:1203.1895v3** **[cs.CC]** for this version)

Submission history

From: Alan Guo [view email]

[v1] Thu, 8 Mar 2012 19:37:20 GMT (627kb,D)

[v2] Thu, 6 Feb 2014 18:24:15 GMT (3330kb,D)

[v3] Sun, 8 Feb 2015 19:45:26 GMT (3425kb,D)

Which authors of this paper are endorsers? | Disable MathJax (What is MathJax?)

Link back to: arXiv, form interface, contact.

Dow
- PD
- Oth
(license)

Curre
cs.CC
< prev
new | r

Chan
cs
cs.(

Refer
- NA

6 blog

DBLF
listi
Gre
Erik
Ala

Book

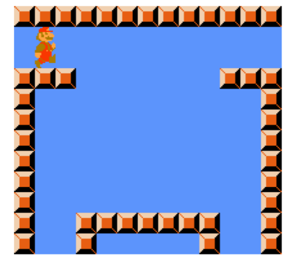


Figure 10: Variable gadget for Super Mario Bros.

shes until it is collected by Mario.

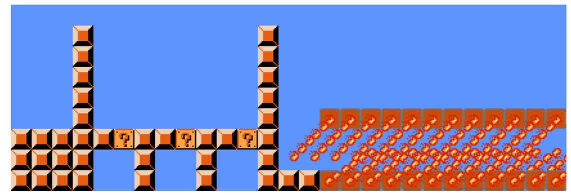


Figure 11: Clause gadget for Super Mario Bros.

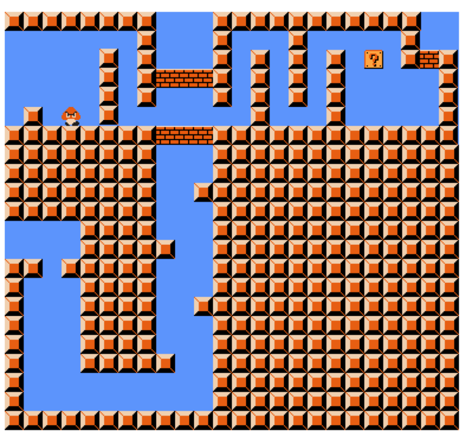
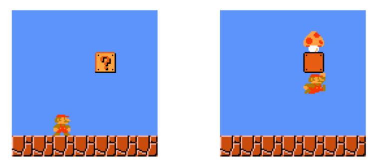


Figure 12: Crossover gadget for Super Mario Bros.



Left: Start gadget for Super Mario Bros. Right: The item block contains a

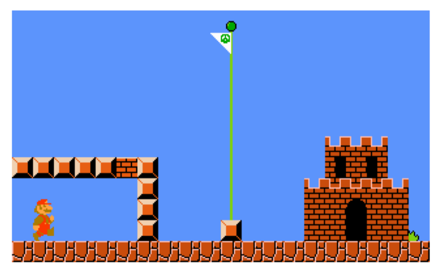
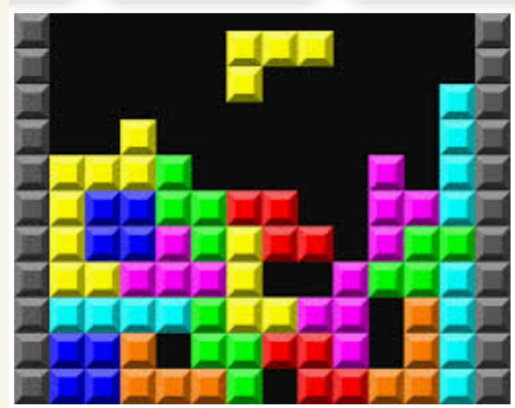


Figure 9: Finish gadget for Super Mario Bros.

Another: Tetris



NP-Hard: Reduce 3-partition

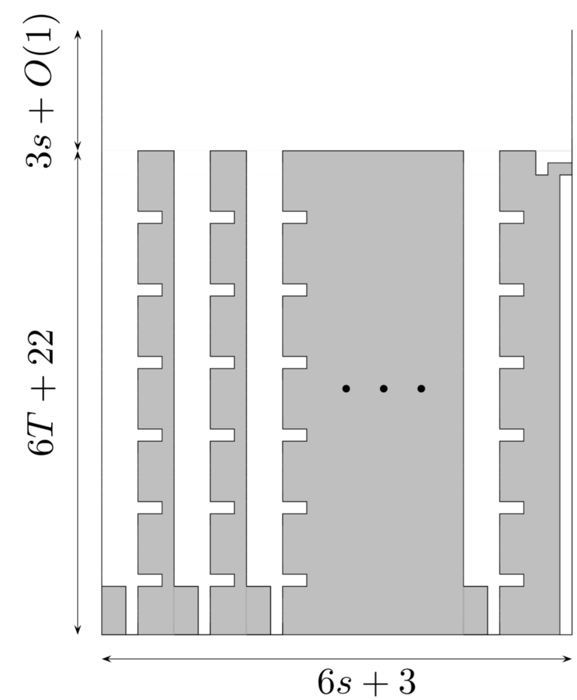


Fig. 2. *The initial gameboard for a Tetris game mapped from an instance of* 3-PARTITION.

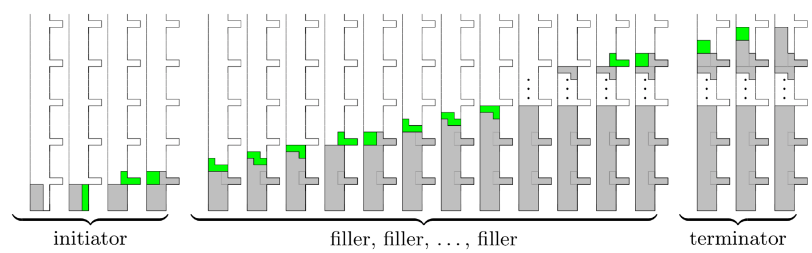


Fig. 3. *A valid sequence of moves within a bucket.*

Again: An active area of research!

arXiv.org > cs > arXiv:1711.00788

Search...

Help | Adva

Computer Science > Computational Geometry

On the complexity of optimal homotopies

Erin Wolf Chambers, Arnaud de Mesmay, Tim Ophelders

(Submitted on 2 Nov 2017)

In this article, we provide new structural results and algorithms for the Homotopy Height problem. In broad terms, this problem quantifies how much a curve on a surface needs to be stretched to sweep continuously between two positions. More precisely, given two homotopic curves γ_1 and γ_2 on a combinatorial (say, triangulated) surface, we investigate the problem of computing a homotopy between γ_1 and γ_2 where the length of the longest intermediate curve is minimized. Such optimal homotopies are relevant for a wide range of purposes, from very theoretical questions in quantitative homotopy theory to more practical applications such as similarity measures on meshes and graph searching problems.

We prove that Homotopy Height is in the complexity class NP, and the corresponding exponential algorithm is the best one known for this problem. This result builds on a structural theorem on monotonicity of optimal homotopies, which is proved in a companion paper. Then we show that this problem encompasses the Homotopic Fréchet distance problem which we therefore also establish to be in NP, answering a question which has previously been considered in several different settings. We also provide an O(log n)-approximation algorithm for Homotopy Height on surfaces by adapting an earlier algorithm of Har-Peled, Nayyeri, Salvatipour and Sidiropoulos in the planar setting.

Almost any problem in AI is NP-Hard.
Not impossible! Just exponential to solve perfectly:
- heuristics
- approximation
- pruning strategies

Linear program

In a linear program, we are given a set of variables

The goal is to give these real values so that:

1. We satisfy some set of linear equations or inequalities

2. We maximize or minimize some linear objective function

Example LP: Cargo plane:

- can carry 100 tons, + volume of 60 cubic meters
- 3 materials:
  - 1st: 2 tons/cubic meter, 40 cubic meters available, worth $1000 per c.m
  - 2nd: 1 ton/c.m, 30 cm total available, and worth $1200 per cm
  - 3rd: 3 tons/cm, 20 cms total, & $12,000 per c.m

Profit!

maximize

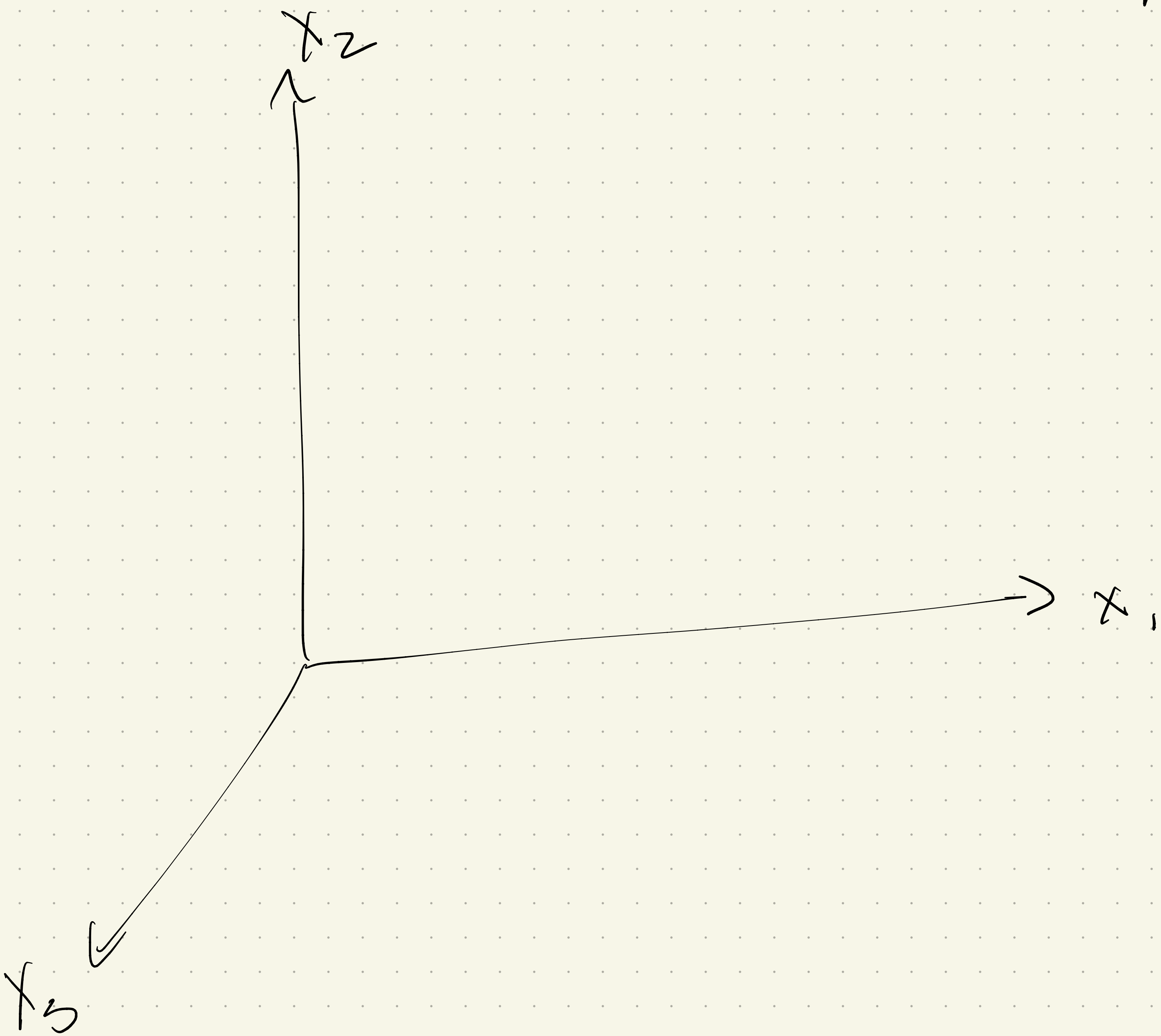
such that:

$x_1 \leq$

$x_2 \leq$

$x_3 \leq$

Geometry: Each equation makes a plane (since linear!):



Each variable adds a dimension:

Maximize $x_1 + 6x_2 + 13x_3$

s.t.

$x_1 \le 200$ (1)

$x_2 \le 300$ (2)

$x_1 + x_2 + x_3 \le 400$

$x_2 + 3x_3 \le 600$

and

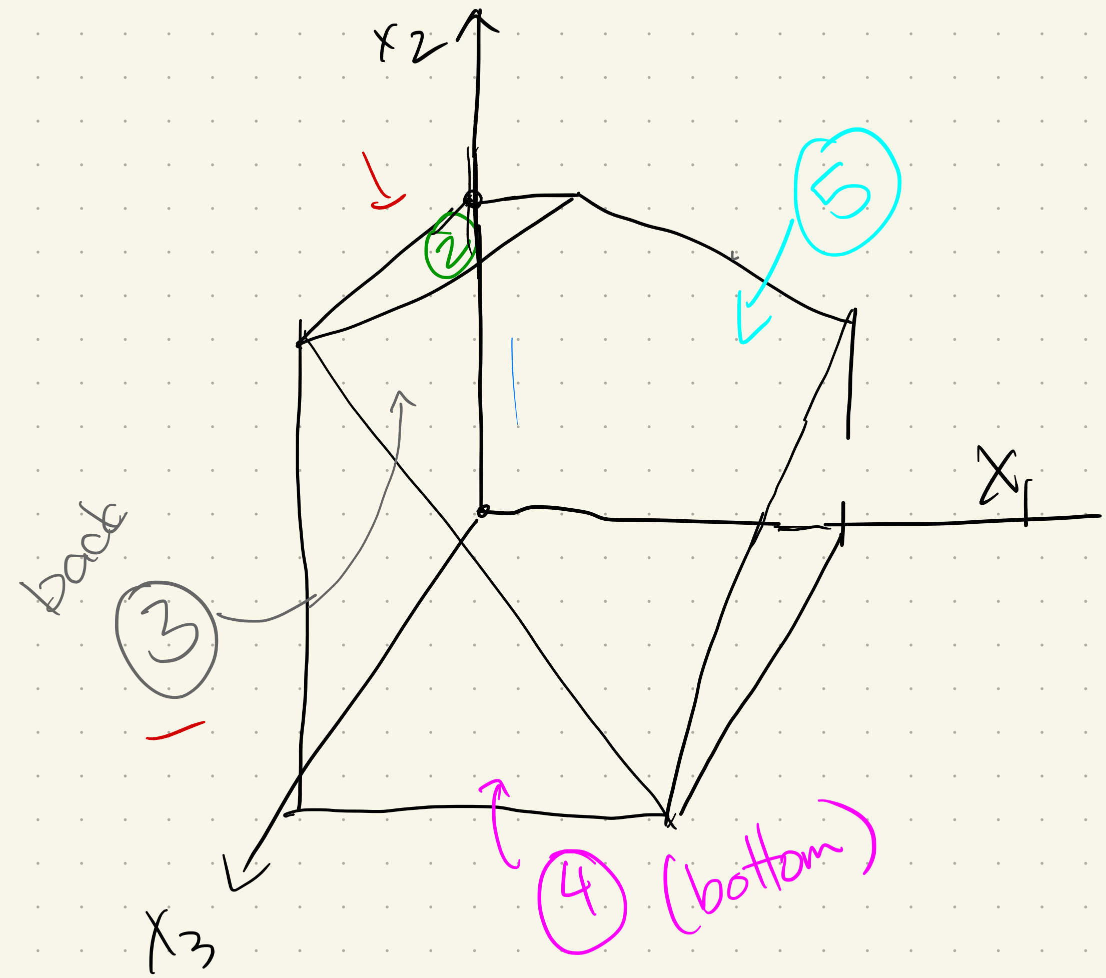
$x_1 \ge 0$ (3)

$x_2 \ge 0$ (4)

$x_3 \ge 0$ (5)

$\mathbb{R}^3$:

And each eqn adds a face to polyhedron:



Another (more general)

n foods, m nutrients

Let a_{ij} = amount of nutrient i in food j

r_i = requirement of nutrient i

x_j = amount of food j purchased

c_j = cost of food j

Goal: Buy food so you satisfy nutrients while minimizing cost

Can view as matrix ⟶

$$A = \begin{bmatrix} & j & \\ i & a_{ij} & \\ & & \end{bmatrix}$$

$$\vec{r} = (r_1, r_2, \ldots r_m)$$

$$\vec{x} = (x_1, \ldots, x_n)$$

$$\vec{c} = (c_1, \ldots, c_n)$$

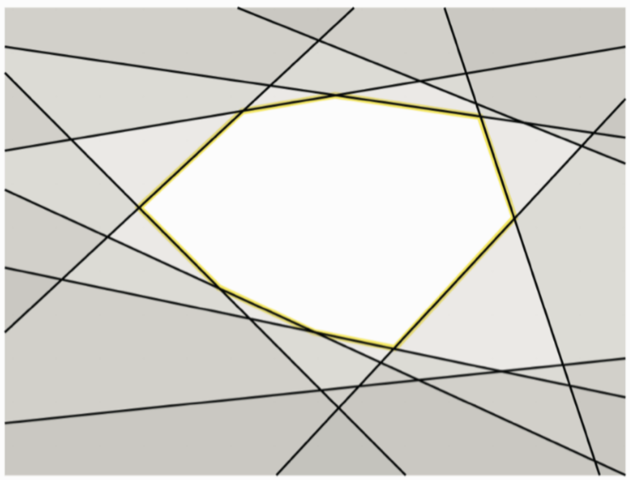
So: minimize

s.t.

In general, get systems like this:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{d} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{d} a_{ij} x_j \le b_i \quad \text{for each } i = 1 \,..\, p \\ & \sum_{j=1}^{d} a_{ij} x_j = b_i \quad \text{for each } i = p+1 \,..\, p+q \\ & \sum_{j=1}^{d} a_{ij} x_j \ge b_i \quad \text{for each } i = p+q+1 \,..\, n \end{aligned}$$

Geometric picture:



A two-dimensional polyhedron (white) defined by 10 linear inequalities.

Canonical form:

Avoid having both $\leq$ and $\geq$.

Why?

So get something more like our first example:

$$\text{maximize} \sum_{j=1}^{d} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad \text{for each } i = 1..n$$

$$x_j \geq 0 \quad \text{for each } j = 1..d$$

Or, given a vector $\vec{c}$, matrix A + vector $\vec{b}$:

Anything can be put into canonical form:

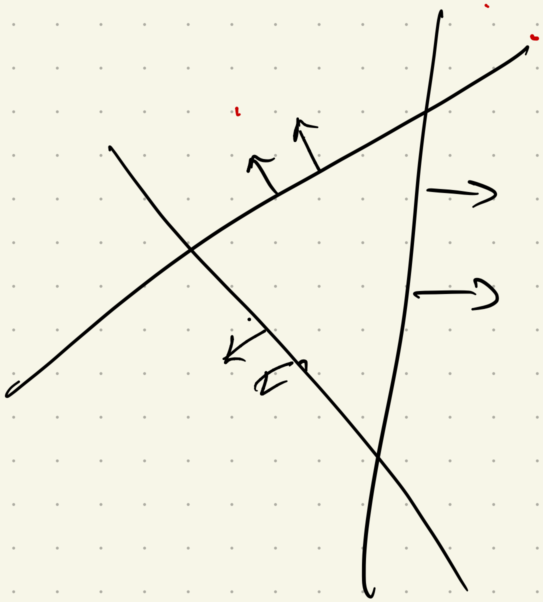
The reduction:

① Avoid $=$:

② Avoid $\geq$:

How could these <u>not</u> have a solution?

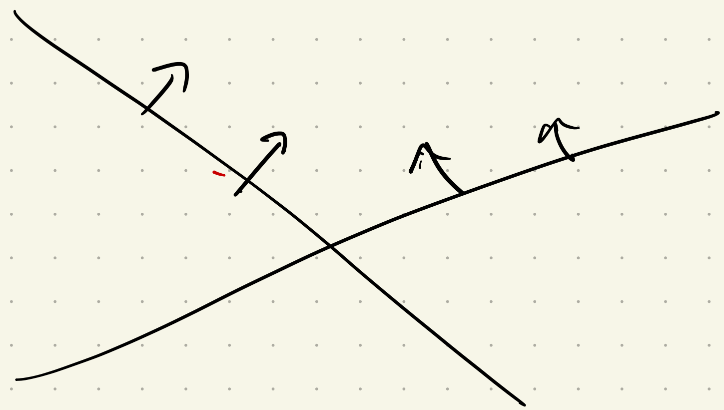
2 ways:



or

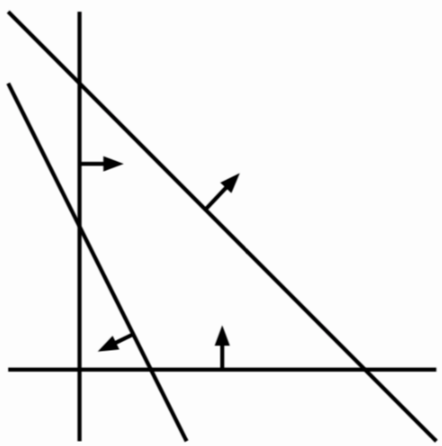
maximize $x_1 + x_2$

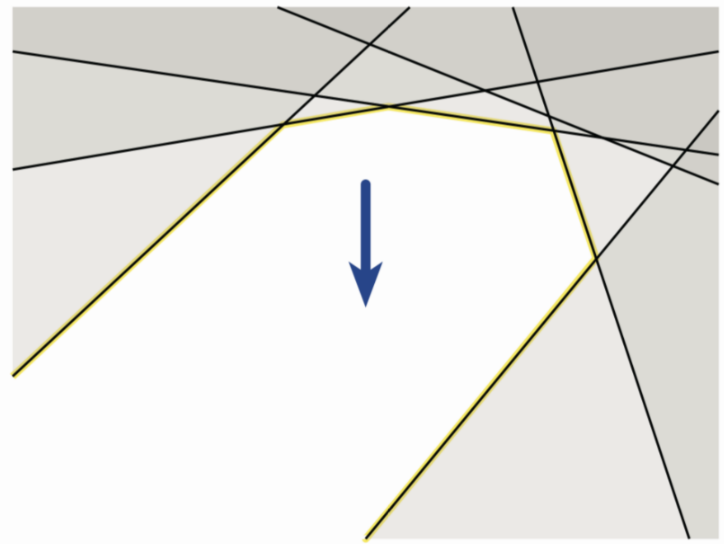
s.t.



Better pictures (still 2-d):

$$\begin{aligned} &\text{maximize } x - y \\ &\text{subject to } 2x + y \leq 1 \\ &\qquad\qquad\quad x + y \geq 2 \\ &\qquad\qquad\quad x, y \geq 0 \end{aligned}$$





Note:

① Multiplying by -1 turns any maximization problem into a minimization one:

why?

② Can turn inequalities into equalities via slack variables:

$$\sum_{i=0}^{n} a_i x_i \leq b \implies$$

(3) Can change equalities into inequalities, also!

$$\sum_{i=1}^{n} a_i x_i = b$$

$$\Downarrow$$

The algorithm: Simplex

Assumes canonical form:

$$\text{maximize} \sum_{j=1}^{d} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad \text{for each } i = 1 \,..\, n$$

$$x_j \geq 0 \quad \text{for each } j = 1 \,..\, d$$

So:

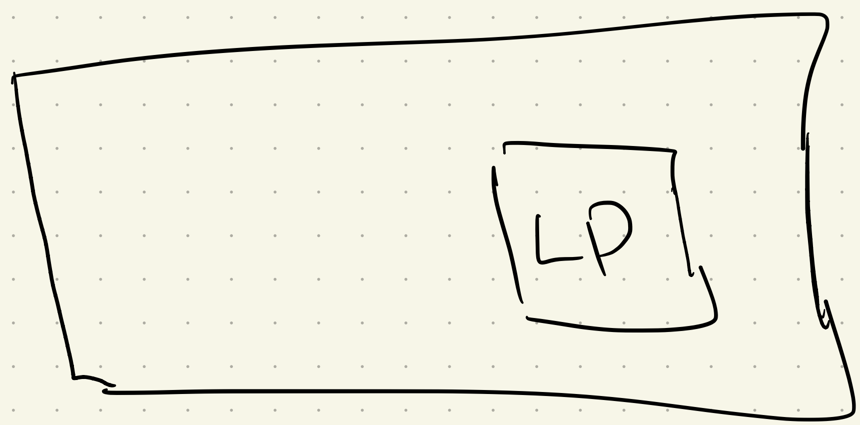
- no min
- only $\leq$
- $x \geq 0$ for all variables

How to implement, plus runtime:

Connections to other problems:

It turns out that LPs are powerful enough to express <u>many</u> types of problems.

In a sense, we solve many problems by <u>reducing</u> them to an LP:



Ex: Flows & Cuts

Input: directed G w/ edge capacities $c(e)$ & $s, t \in V$

Goal: Compute flow $f: E \to \mathbb{R}$ s.t.

1. $0 \leq f(e) \leq c(e)$
2. $\forall v \neq s, t$,

$$\sum_{u} f(u \to v) = \sum_{w} f(v \to w)$$

Make an LP: Maximize

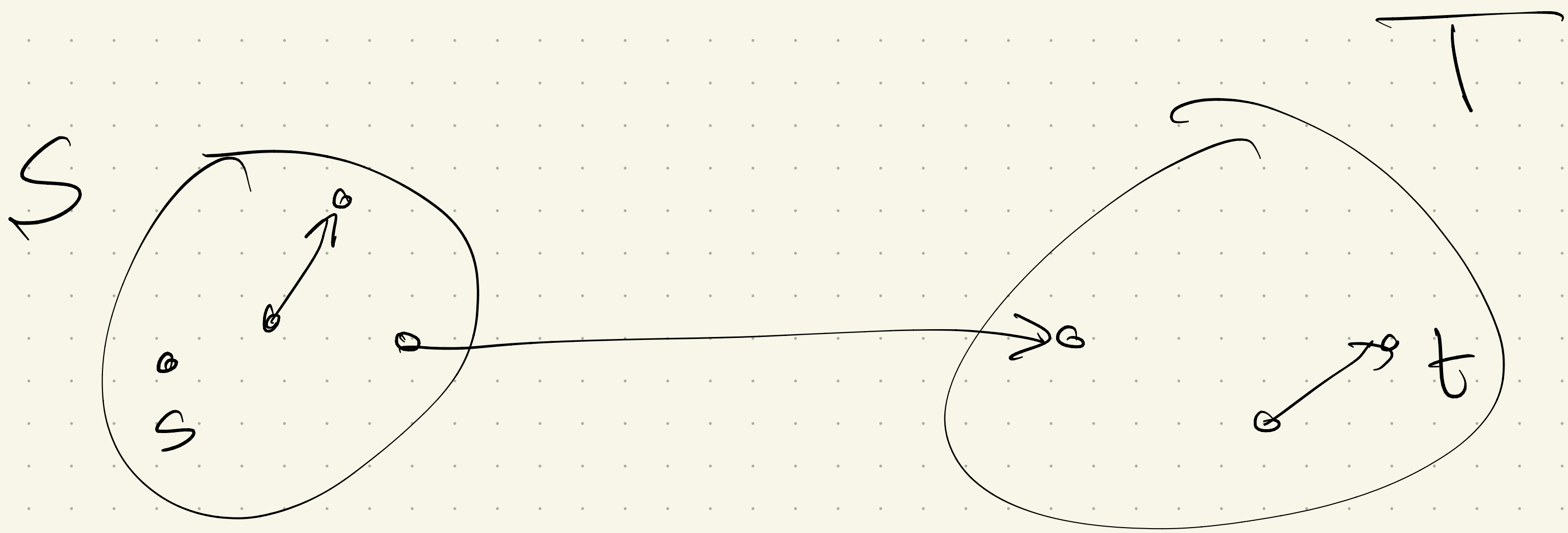
s.t.

Related: Min cuts (S,T)

Use indicator variables:

$$s_v = 0 \text{ or } 1$$

$$x_e = x_{(u \to v)} = 1 \text{ if } u \in S \text{ and } v \in T$$



The LP:

Minimize $\sum_{u \to v} C_{u \to u} \cdot X_{u \to v}$ ← want few edges

s.t.

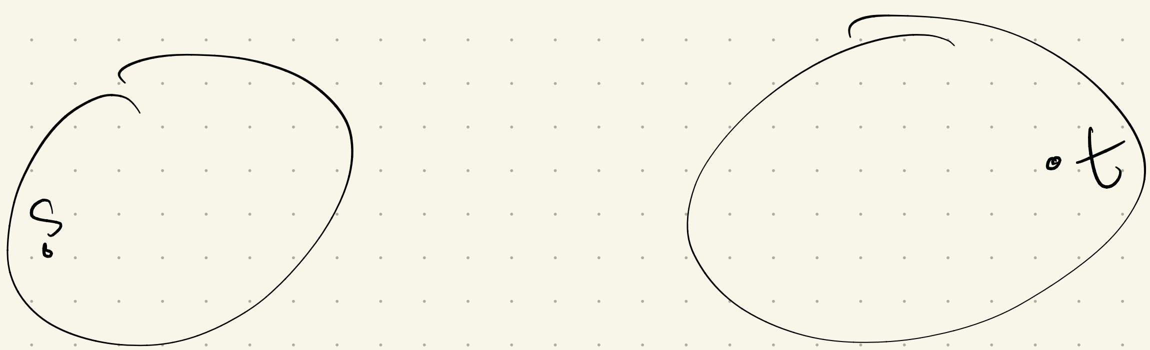
$$X_{u \to v} + S_v - S_u \geq 0 \quad \forall u \to v$$

← Which are forced?

$$X_{u \to v} \geq 0 \quad \forall u, v$$

$$S_s = 1$$

$$S_t = 0$$



Note:

For flow/cuts, a solution would yield optimal LP solution.

The reverse is not obvious!

LP might have strange fractional answer which doesn't describe a cut.

It can be shown that this won't happen

↳ but not obvious...