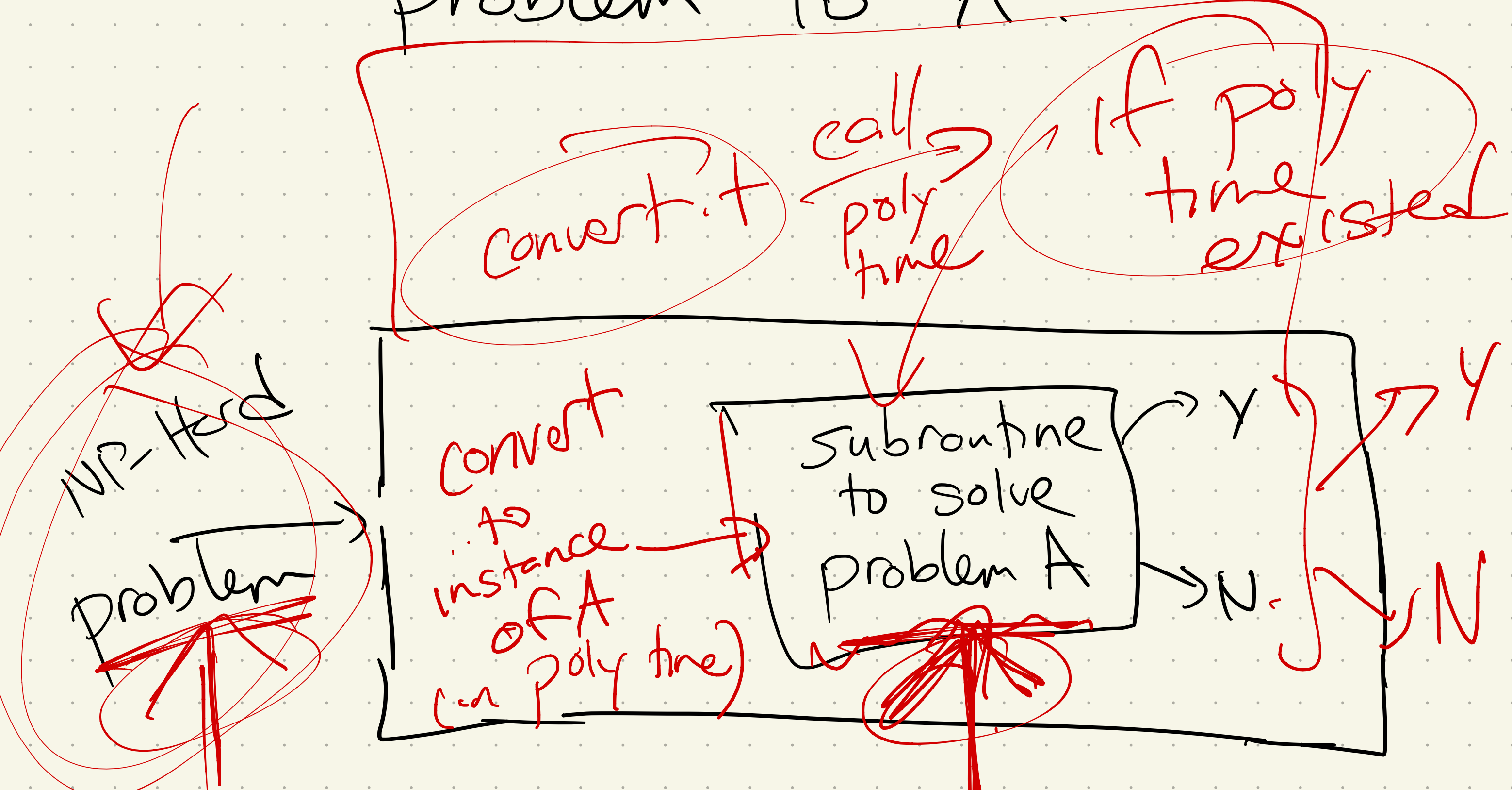
Algorithms 2020

NP-Hardness:
Last day!

- Lecture notes are now updated
- Oral grading: Thurs. + Fri.
- Final HW: due Wed. before Thanksgiving
- Practice MT + final - up later today

To prove NP-Hardness of A:

Reduce a known NP-Hard problem to A.



general instance of known problem

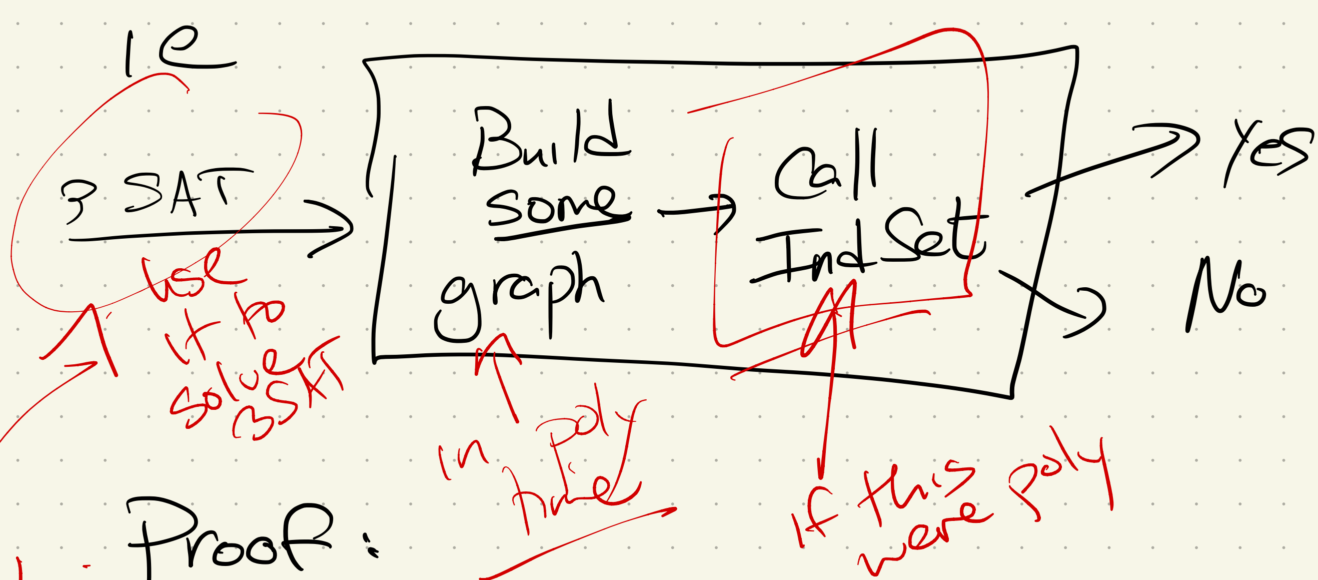
many of these

normally: to solve X, reduce to A & call subroutine

The Pattern: Reductions

1) Find an NP-Hard problem, & solve it using unknown problem as a subroutine

ie



Key: Proof:

Need if & only if!

(ie might be some wierd indep. set that doesn't make a SAT)

Challenge: Finding correct NP Hard problem

So far:

- CircuitSAT
- SAT
- 3SAT

logic

- Ind. Set
- Clique
- Vertex Cover
- 3-Coloring

graphs

- Subset Sum

#one

- Set Cover

Subset Sum:

Given a set of numbers $X = \{x_1, x_2, x_3, \ldots, x_n\}$ I
and a target t, does some subset of X sum to t?

$\log t$ bits in input

Ex:

(actually did this one!
See lecture from Ch. 2

Runtime:

backtracking:
every # is either in set, or not

n #s

exponential

use DP: $O(nT)$

memoize!

size T

Subset Sum is NP-Hard.

Reduction: Vertex Cover

Input: Graph G & size k

Challenge: Need to construct a set of numbers, so that we hit some target sum if & only if a vertex cover in G of size k exists.

Recall: Base 4

$31203 = 3 \cdot 4^0 + 0 \cdot 4^1 + 2 \cdot 4^2 + 1 \cdot 4^3 + 3 \cdot 4^4$

$= \#$

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

"gadget"

Number edges $0 \ldots E-1$ (edge)

↳ create a number for subset sum: E #s

e_0: $b_0 = 1 = 4^0 = (\underbrace{0 \cdots 0}_{E-1}1)_4$

e_1: $b_1 = 4^1 = 4 = (0 \cdots 010)_4$

e_2: $b_2 = 4^2 = 16 = (0 \cdots 100)_4$

e_3: $b_3 = 4^3 = (1000)_4$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

e_{E-1}: $b_{E-1} = 4^{E-1} = (1\underbrace{0 \cdots 0}_{E-2})$

For each vertex, make another #:

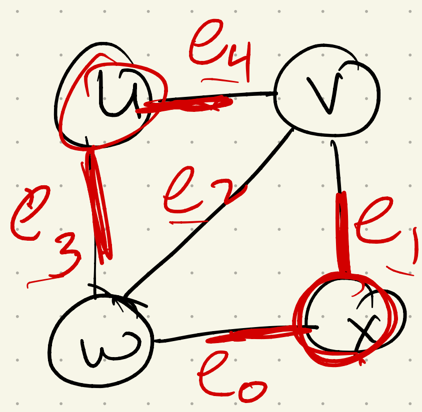
$$a_v := \underbrace{4^E}_{> \text{ any edge } \# \; b_i} + \underbrace{\sum_{e_i \text{ into/out of } v} 4^i}$$

$(1 \underbrace{\qquad d(v) \text{ 1's here} \qquad}_{})$

E ↑ 0 if edge 4 is not connected to v
1 if it is

Think of base 4 representation:

4^E spot



4^E 4^3 4^4

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

vertex #s

edge #s

$(\underset{E}{1}00011)_4$

$(V+E)$ #s

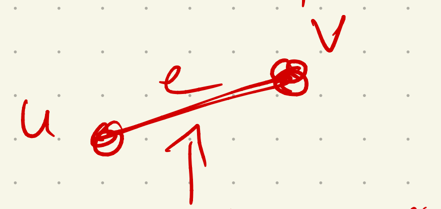
Now, set $T = \boxed{k \cdot 4^E} + \sum_{i=0}^{E-1} 2 \cdot 4^i$

Why?

$k \cdot 4^E$ is big - if only get k vertices, the only way to get $\geq T$ is to pick k of vertex #s.

only 1 edges #s + 2 vertex #s w/ 1's in lower spot

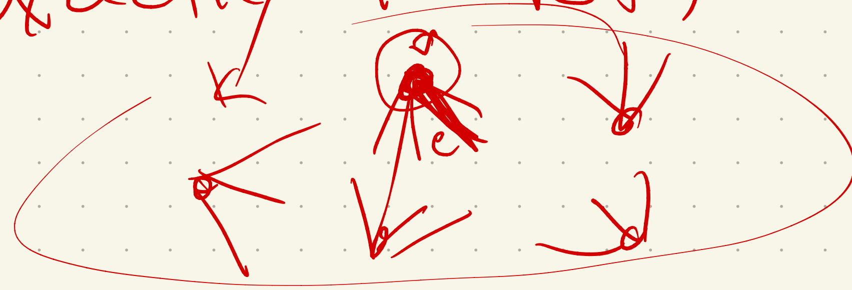
$E \quad \underbrace{E-1 \; \cdots \; 0}$



e^{th} edge # + 2 vertex #s

any sum of #s has ≤ 3 in lower places

In order to hit target, I need exactly k vertex numbers.



<u>Proof</u>: size k VC $\Longleftrightarrow$ sum to T

$\Rightarrow$: VC: k vertices covered all edges

Then #s corresponding sum to $k \cdot 4^E$ (for top term)

$+ \sum_{\text{edges covered}} 4^e \rightarrow$ not quite T

However, add edges to get sum $= \sum_{\text{edge } i} 2 \cdot 4^i$

$\rightarrow$ k a_i's + E e_i's numbers

$\Leftarrow$: subset sum = T

some a_i's + b_i's = T

must have = k a_i's + cover all e's + included e_i #s

$\Leftarrow$: take the a_i's + b_i's that

$$= \underbrace{k\cdot 4^E}_{k\ a_i\text{'s}} + \underbrace{\sum_{i=0}^{E-1} 2\cdot 4^i}$$

all e #s sum to

$$\underset{E}{0}\ 1\ 1 \cdots 1\ \underset{0}{1}$$

so must have another $1\cdot 4^i$ from a #s

$\Rightarrow$ k vertices cover all edges

Time to reduce:

$\left.\begin{array}{l} V+E \text{ \#S} \\ \& \text{ a } T \end{array}\right\} O(V+E)$

So: If I could solve Subset Sum in poly time, I could use it to solve VC.

$\Rightarrow$ Subset Sum is also NP-Hard

(& since in NP, also NP-Complete)

Set Cover:

Given a set U of n elements, a collection $S_1, S_2, \dots, S_m$ of subsets of U, + a number k, is there a collection of k of the S_i's whose union is all of U?

$\binom{m}{k} \approx m^k$

$u_1 \dots u_{13}$

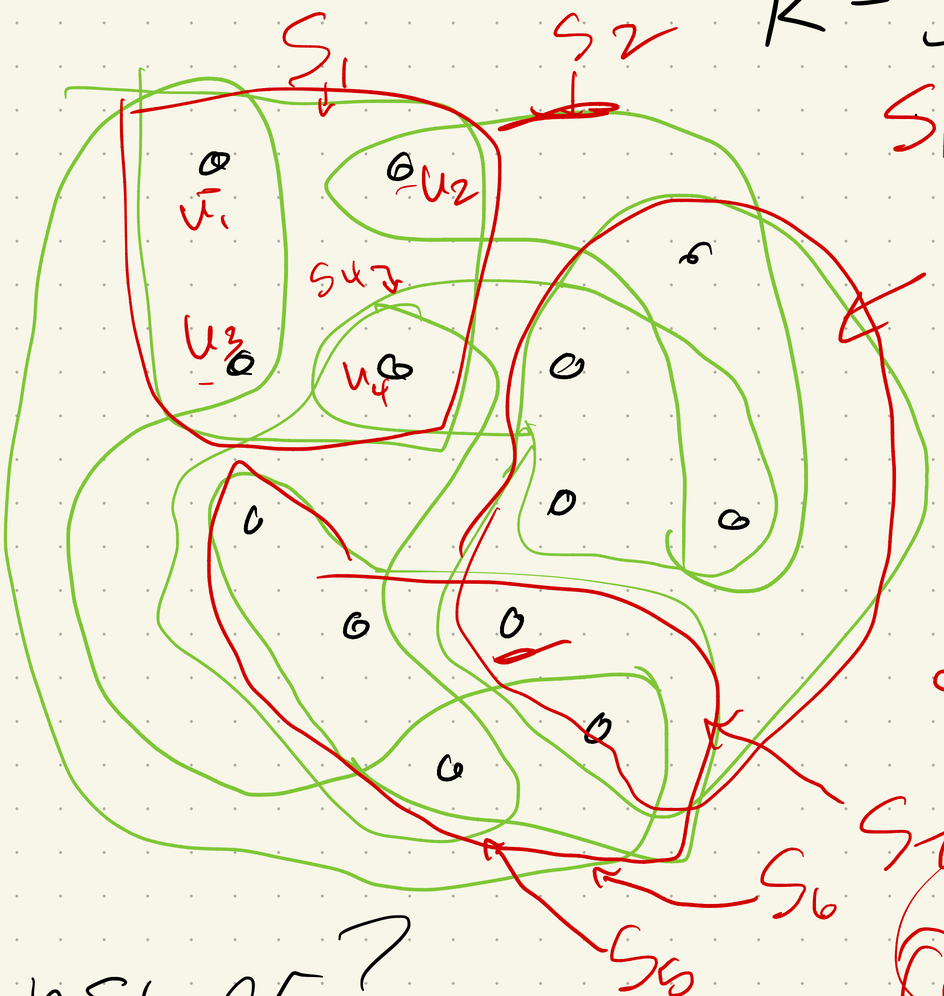
$K = 3$?

Ex:

elements in U:

Subsets $S_1, \dots, S_7$

$m = 7$



$S_1 = \{u_1, u_2, u_3, u_4\}$

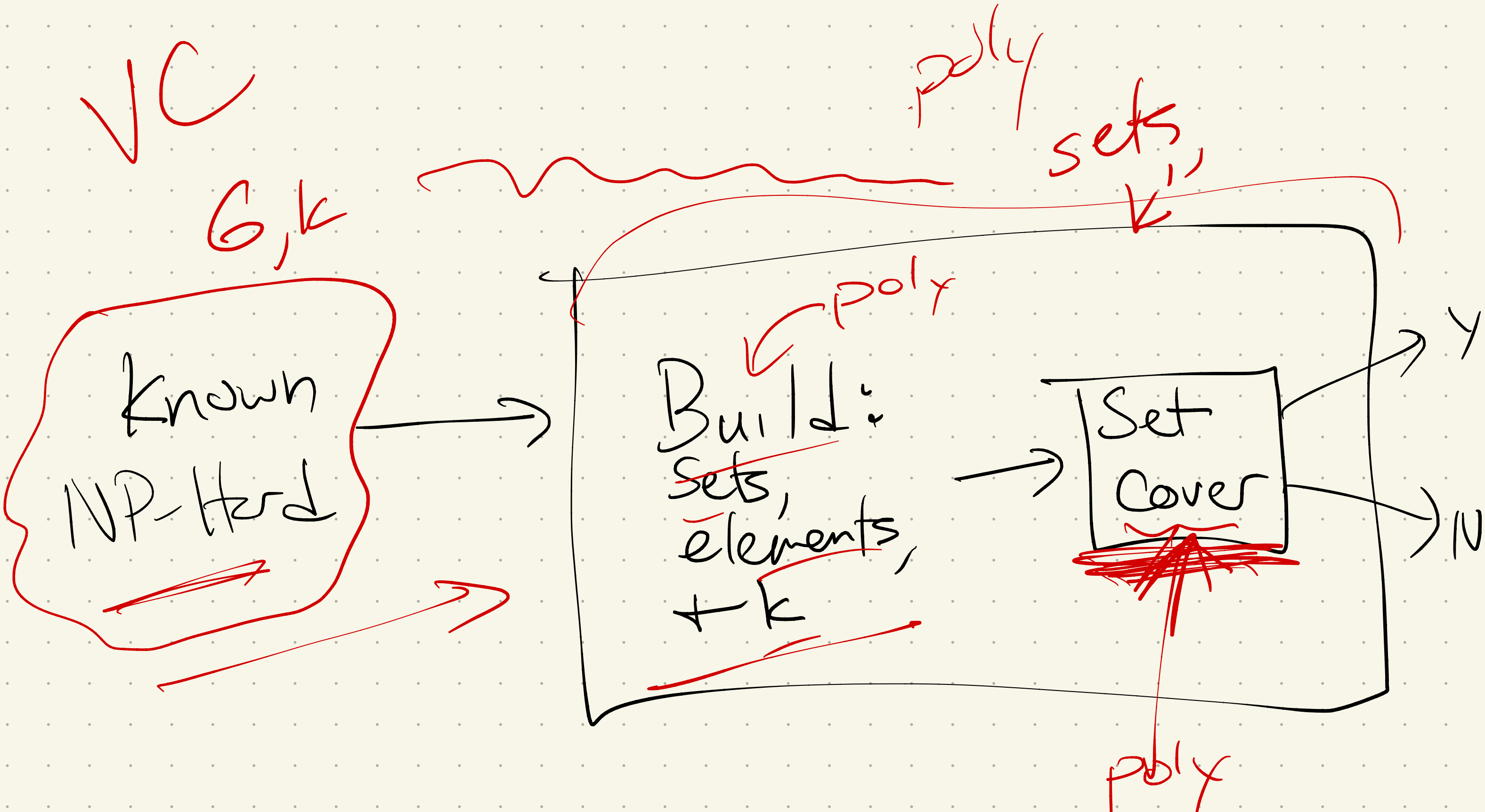
Yes!

$S_3, S_1,$ & S_5

Answer?

Set Cover is NP-Hard:

Need to reduce!



So: if I could solve Set cover in poly time, could solve some known NP-Hard problem.

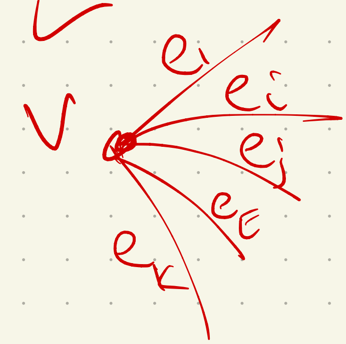
cover edges

Reduction from vertex cover,
so input is G & K.
(V,E)

Construct:

$\underline{U}$ = each item an edge
$X_1, \dots X_E$

$\underline{S_i}$'s : make a set per vertex
S_v contain all elements
which v is incident to



$S_v = \{X_1, X_i, X_j, X_E, X_k\}$

& $\underline{k}$: K same

Vertex cover of size k

$\Longleftrightarrow$ $\boxed{\text{Set cover of size } k}$

$\Rightarrow$ if vertex cover of size k:
take those k sets in set cover

since all edges in G, are covered, & U = V, all ~~items~~ in U are covered

$\Leftarrow$: Spps k set cover then corresponds to k vertices, & all elements in U were covered, same vertex set covers edges

Some fun examples

arXiv.org > cs > arXiv:1203.1895

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

(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.

Dov
- PD
- Otl

(license)

Curre
cs.CC
< prev
new | r

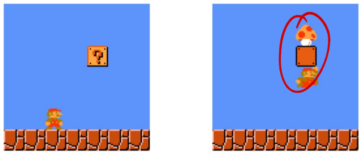
Chan
cs
cs.(

Refer
- NA

6 blog

DBLF
listi
Gre
Eril
Ala

Booki



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

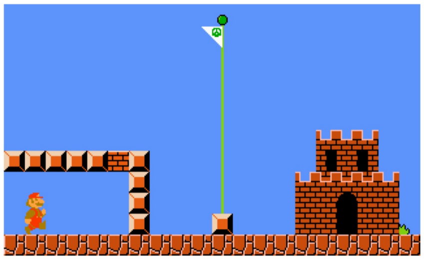


Figure 9: Finish gadget for Super Mario Bros.

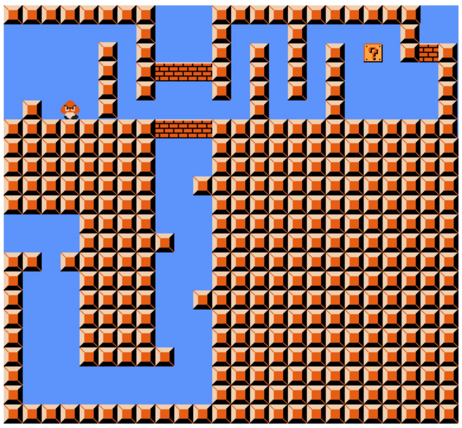


Figure 12: Crossover gadget for Super Mario Bros.

iPad 10:44 AM arxiv.org 54%

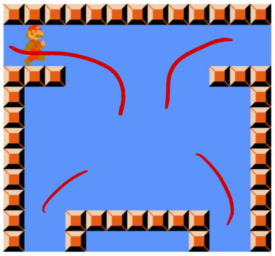


Figure 10: Variable gadget for Super Mario Bros.

shes until it is collected by Mario.

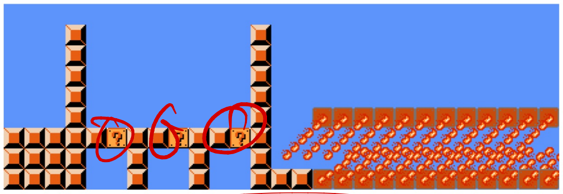
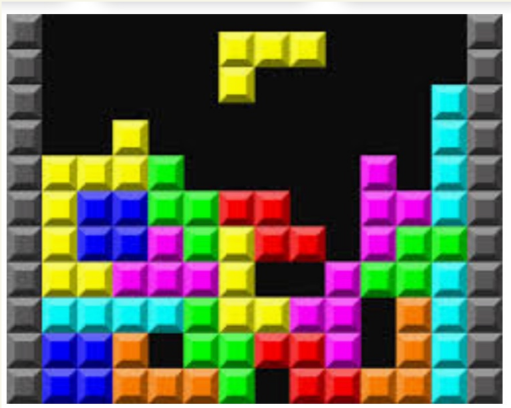


Figure 11: Clause gadget for Super Mario Bros.

Another: Tetris



NP-Hard:

Reduce 3-partition

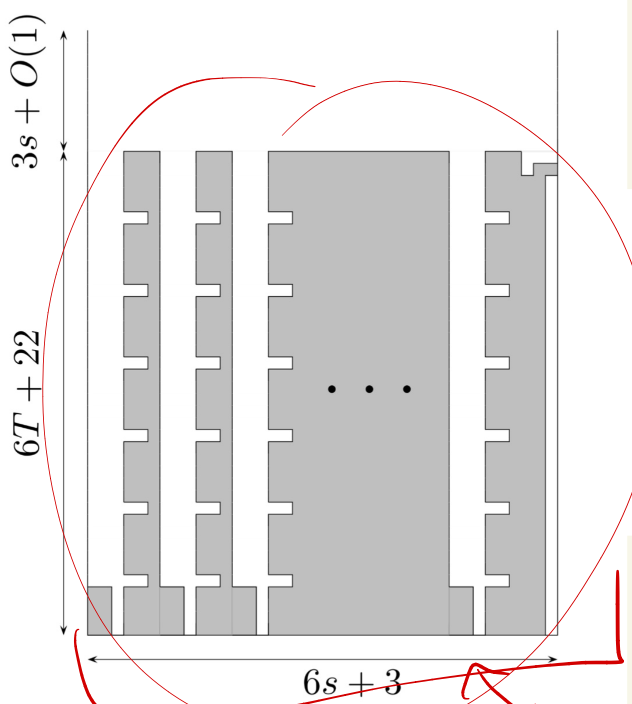


Fig. 2. *The initial gameboard for a Tetris game mapped from an instance of 3-*PARTITION.

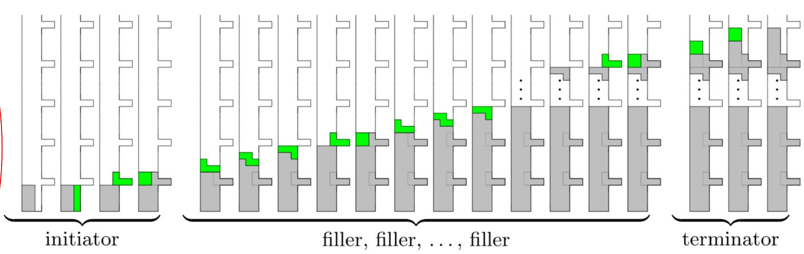


Fig. 3. *A valid sequence of moves within a bucket.*

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.

OS

Almost any problem in AI is NP-Hard.

Not impossible! Just exponential to solve perfectly:

- heuristics
- approximation
- pruning strategies

Friday

- Finally on to LP :)

Monday

- No class

Resume on Wed.