Computational Models - Lecture 12[1]

Handout Mode

Iftach Haitner and Yishay Mansour.

Tel Aviv University.

May 26/28, 2014

[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- Reminder - poly-time reductions, NP-completeness and SAT
- SAT is NP-Complete
- SAT $\leq_P$ 3SAT
- Hamiltonian Path Is NP-Complete
- SUBSET-SUM Is NP-Complete
- Integer-Programming is NP-Complete

- Sipser, 7.4–7.5

Section 1

Reminder

Reminder – Poly-time Reductions

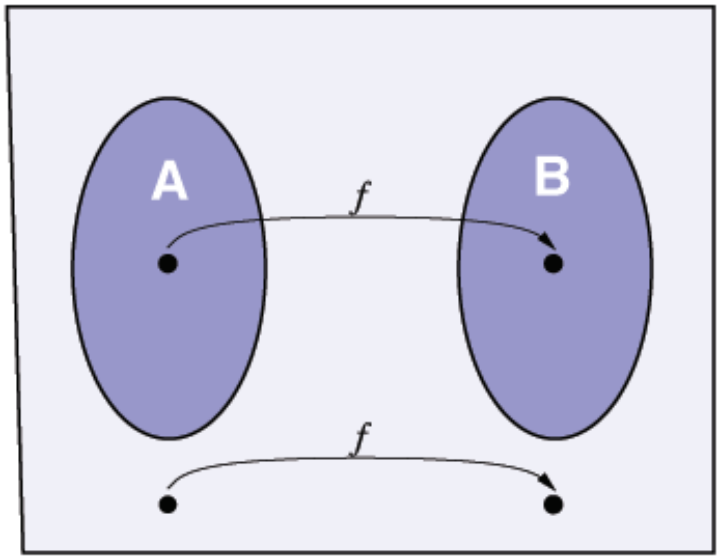
Definition 1

A language A is poly-time mapping reducible to B, denoted $A \leq_P B$, if exists poly-time computable function f such that

$$w \in A \iff f(w) \in B.$$

for every $w \in \Sigma^*$.
The function f is called a polynomial-time reduction from A to B.

The mapping f efficiently converts questions about membership in A to membership in B.



Reminder – NP Completeness

Definition 2 (NP-complete)

A language B is NP-complete, if

- $B \in \mathcal{NP}$, and
- Every $A \in \mathcal{NP}$ is poly-time reducible to B

We let $\mathcal{NPC}$ denote the class of all NP-complete languages.

$$A_{NP} = \{\langle M, x, 1^n \rangle : M \text{ is a Turing Machine} \wedge \exists c \in \Sigma^* \text{ s.t. } M(x, c) \text{ accepts within } n \text{ steps}\}$$

Theorem 3

$A_{NP} \in \mathcal{NPC}$

Theorem 4

Assume that $B \in \mathcal{NP}$, $A \in \mathcal{NPC}$ *and* $A \leq_P B$, *then* $B \in \mathcal{NPC}$.

Boolean Formulas and SAT

A Boolean formula is an expression involving Boolean variables and operations.

$$\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$$

Definition 5 (satisfiable formula)

A formula is satisfiable, if some assignment of 0s and 1s to the variables makes the formula evaluate to 1.

The formula $\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ is satisfiable by the assignment

$$\begin{aligned} x &= 0 \\ y &= 1 \\ z &= 0 \end{aligned}$$

The language of satisfied formulas:

$$\text{SAT} = \{\langle\phi\rangle : \ \phi \text{ is a satisfiable Boolean formula}\}$$

Section 2

SAT is NP-Complete

Theorem 6 (Cook-Levin, early 70s)

SAT $\in \mathcal{NPC}$.

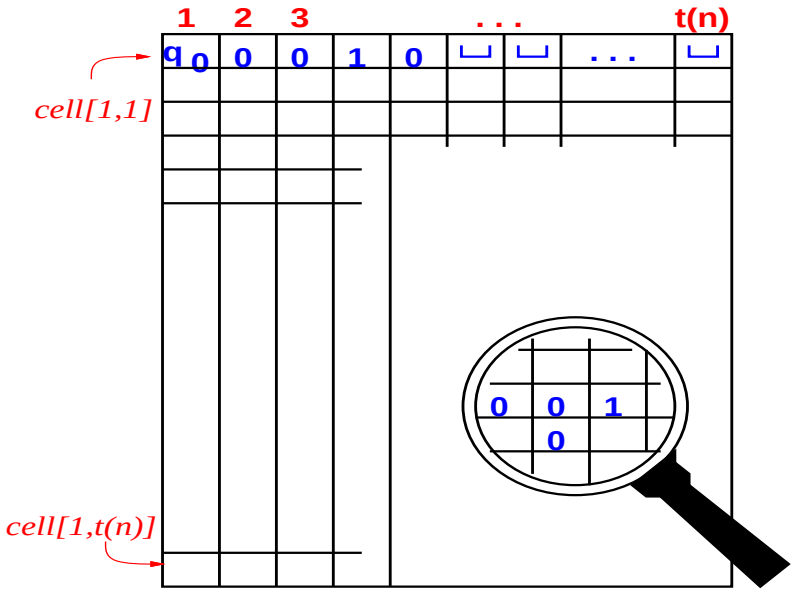
Proof's idea:

- Suppose L $\in \mathcal{NP}$ and let M $= (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be an n^c-time NTM that accepts L.
- Given the string $w \in \{0,1\}^*$, we construct in time $O(|w|^{2c})$ a formula $\phi_{M,w}$ such that: $\phi_{M,w} \in$ SAT iff M accepts w
- Hence, the mapping $w \mapsto \phi_{M,w}$ is a poly-time reduction from L to SAT, establishing L $\leq_P$ SAT.

- In the following we fix L, M and $w \in \{0,1\}^n$.

The Configuration-History Tableau

Consider the n^c-by-n^c tableau that describes a computation history of M on input w.



- First row represents initial configuration of M on input w.
- i'th row represents a possible i-th configuration in a computation of M on input w.

The Boolean formula $\phi_{M,w}$ "mimics" the tableau.

The formula $\phi_{M,w}$

- Let $S = Q \cup \Gamma$ be the alphabet of the configuration history
- $\phi_{M,w}$ uses Boolean variables $\{x_{i,j,s}\}_{i,j \in [n^c], s \in S}$:
  Idea: $x_{i,j,s} = 1$ iff the j-th cell in i'th configuration contains the symbol s:
- The formula $\phi_{M,w}$ is of the from:

$$\phi_{M,w} = \phi_{\text{Cell(M)}} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Accept(M)}} \wedge \phi_{\text{Move(M)}}$$

- Given an assignment z for $\phi_{M,w}$, let $T(z)$ be the $n^c \times n^c$ tableau, defined by setting the j-th cell in i'th configuration to s, if $x_{i,j,s} = 1$ in z.
  ($T(x)$ is undefined. in case $x_{i,j,s'} = x_{i,j,s} = 1$ for some $s \neq s' \in S$, or $x_{i,j,s'} = 0$ for all $s \in S$).

The formula $\phi_{\text{Cell(M)}}$

$\phi_{\text{Cell(M)}}$ guarantees that the variables encode legal configurations:

- Each cell (i,j) has at least one letter: $\bigvee_{s\in S} x_{i,j,s}$.
- No cell (i,j) has two or more letters $\bigwedge_{s\neq s'\in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}}$.

  Together:

$$\phi_{\text{Cell(M)}} = \bigwedge_{i,j} \left[\left(\bigvee_{s\in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s\neq s'\in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}} \right) \right]$$

Claim 7

Assume z satisfies $\phi_{\text{Cell(M)}}$, then $T(z)$ is defined.

The formula $\phi_{\text{Start}(w)}$

$\phi_{\text{Start}(w)}$ guarantees that the first row encodes the initial configuration (i.e., $q_0 w$).

$$\phi_{\text{start}(w)} = x_{1,1,q_0} \wedge x_{1,2,w_1} \wedge x_{1,3,w_2} \wedge \ldots \wedge x_{1,n+1,w_n} \wedge x_{1,n+2,\sqcup} \wedge \ldots \wedge x_{1,n^c,\sqcup}$$

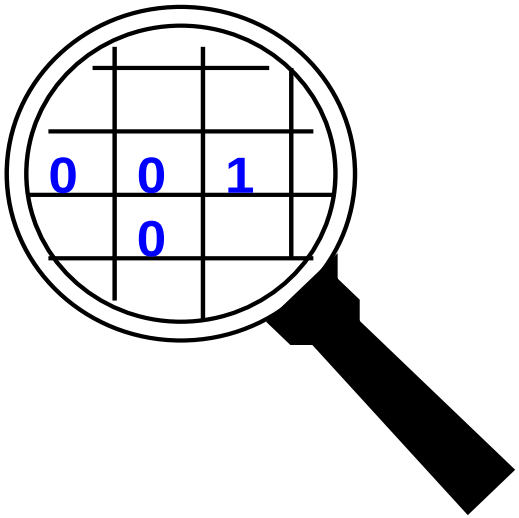
Claim 8

Assume z satisfies $\phi_{\text{Cell(M)}} \wedge \phi_{\text{Start}(w)}$, then the first line of $T(z)$ is $q_0 w \underbrace{\sqcup \ldots \sqcup}_{n^c - n - 1}$.

The formula $\phi_{\text{Move(M)}}$

$\phi_{\text{Move(M)}}$ is the "heart" of $\phi_{\text{M},w}$. To construct it, we employ locality of computations.

- To determine content of tableau entry (i,j) (i.e., cell j in configuration i), it only requires to know M's table and the contents of the three tableau entries: $(i-1,j-1),(i-1,j),(i-1,j+1)$.



- If head not in area, nothing changes.
- If head is in area, changes are local and are determined by δ.

$\phi_{\text{Move(M)}}$ – Rectangles

A rectangle is a 2×3 configuration sub-table.

Assume that
$\delta(q_1, a) = \{(q_1, b, \text{R})\}$ and $\delta(q_1, b) = \{(q_2, c, \text{L}), (q_2, a, \text{R})\}$.

- Legal 2×3 rectangles:

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

a	b	a
a	b	q_2

b	b	b
c	b	b

a	a	b
a	a	b

- Illegal 2×3 rectangles:

a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

$\phi_{\text{Move(M)}}$ – Characterizing legal rectangles

The formula “verifies" that all 2×3 rectangles in the Tableau are in the list C:For all $a, a', b, b', c \in \Gamma$, $q, q' \in Q$ and $* \in S$, add the following rectangles to C:

1.

a	b	c
$*$	b	$*$

2.

a	q	b
q’	a	b’

$(\text{L}, q', b') \in \delta(q, b)$

a	q	b
a	b’	q’

$(\text{R}, q', b') \in \delta(q, b)$

3.

q	$*$	$*$
$*$	$*$	$*$

$*$	$*$	q
$*$	$*$	$*$

- Some rectangles in C are clearly illegal.
- For rectangles on the left-most and right-most side of Tableau, we use a slightly different first type rectangles.

$\phi_{\text{Move(M)}}$– formal definition

- For each entry $(i,j) \in [n^c] \times [n^c]$ and $c \in C$, let $\phi_{\text{Move},i,j,c}$ be the formula taking the value 1 iff the 2×3 table of cells in the Tableau whose upper-left corner is (i,j) is c.

  For instance, for entry $(1,1)$ and $c =$

a	q_1	b
q_2	a	c

  ,

  let $\phi_{\text{Move},1,1,c} = x_{1,1,a} \wedge x_{1,2,q_1} \wedge x_{1,3,b} \wedge x_{2,1,q_2} \wedge x_{2,2,a} \wedge x_{2,3,c}$

- Finally, let $\phi_{\text{Move(M)}} = \bigwedge_{(i,j)} \bigvee_{c \in C} \phi_{\text{Move},i,j,c}$.

Claim 9

Assume z satisfies $\phi_{\text{Cell(M)}} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move(M)}}$, then $T(z)$ is a possible configuration history of $\text{M}(w)$.

Proof: By induction on the row index. Base case: z satisfies $\phi_{\text{Cell(M)}} \wedge \phi_{\text{Start}(w)}$. Assume configuration defined in rows $1,\ldots,i$ is possible and head is in cell j. The configuration of rows $1,\ldots,i+1$ is also possible: Cells of indices not in $\{j-1,j,j+1\}$, by first type of rectangles in C. Other cells, by second type rectangles in C. Q: Why de we need the third type of cells?

The formula $\phi_{\text{Accept(M)}}$

$\phi_{\text{Accept(M)}}$ guarantees that some row encodes an accepting configuration (i.e., uq_av):

$$\phi_{\text{Accept(M)}} = \bigvee_{i,j} x_{i,j,q_a}$$

Claim 10

Assume z salsifies $\phi_{\text{M},w} = \phi_{\text{Accept(M)}} \wedge \phi_{\text{Cell(M)}} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move(M)}}$, then $T(z)$ is an accepting configuration history of $\text{M}(w)$.

Correctness of Reduction

- The transformation $w \mapsto \phi_{M,w}$ is computable in time $O(n^{2c})$.
- An assignment satisfying $\phi_{M,w}$, corresponds to an accepting configuration history of $M(w)$.
- An accepting configuration history of $M(w)$ corresponds to an assignment satisfying $\phi_{M,w}$. (?)

  Therefore, M accepts w iff $\phi_{M,w} \in SAT$. ♣.

- For complete details, consult Sipser chapter 7.4.

Section 3

3SAT Is NP-Complete

SAT and 3SAT

- SAT $= \{\langle\phi\rangle :\ \phi$ is a satisfiable Boolean formula$\}$
- A Boolean formula is in conjunctive normal form (CNF) if it consists of clauses, connected with $\wedge$'s.
  For example: $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$
- A Boolean formula is in k-CNF form, if it is a CNF formula, and all clauses have k literals.
- 3SAT $= \{\langle\phi\rangle\ :\ \phi$ is satisfiable 3CNF formula$\}$.
- CSAT $= \{\langle\phi\rangle :\ \phi$ is a satisfiable CNF formula$\}$
- One can slightly modify the proof of Cook-Levin theorem we just seen, to prove that CSAT $\in \mathcal{NPC}$! (take home exercise :-))
- Hence, to prove that 3SAT $\in \mathcal{NPC}$, it suffices to prove that CSAT $\leq_P$ 3SAT.

CSAT $\leq_P$ 3SAT

- The reduction maps CNF formulae to 3CNF ones "clause by clause".
  A clause with d literals is mapped to d clauses, built on the original literals together with $(d-3)$ new ones.
- For example: $\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4 \vee x_8) \longmapsto$
  $\phi_3 = (x_1 \vee \overline{x_2} \vee y_1) \wedge (\overline{y_1} \vee \overline{x_3} \vee y_2) \wedge (\overline{y_2} \vee x_4 \vee x_8)$

Claim 11

ϕ has a satisfying assignment iff ϕ_3 does.

Proof's idea:
$\Leftarrow$ An assignment satisfying ϕ_3 cannot "rely" on new literals alone – at least one original literal must be satisfied.
$\Rightarrow$ An assignment satisfying ϕ makes at least one literal per clause happy. In the "ϕ_3 clause" of this literal the new veriable is under no constraints. This enables propagation to a satisfying assignment that "relies" on new vars alone in rest of ϕ_3 clauses.

SAT $\leq_P$ 3SAT, 2

$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4 \vee x_8) \longmapsto$
$\phi_3 = (x_1 \vee \overline{x_2} \vee y_1) \wedge (\overline{y_1} \vee \overline{x_3} \vee y_2) \wedge (\overline{y_2} \vee x_4 \vee x_8)$

Claim 12

ϕ has a satisfying assignment iff ϕ_3 does.

- Doing the above mapping to each clause of a CNF formula, we get a 3CNF that is satisfied iff the original one is.
- Since this mapping is polynomial time (?), we get CSAT $\leq_P$ 3SAT. ♣.

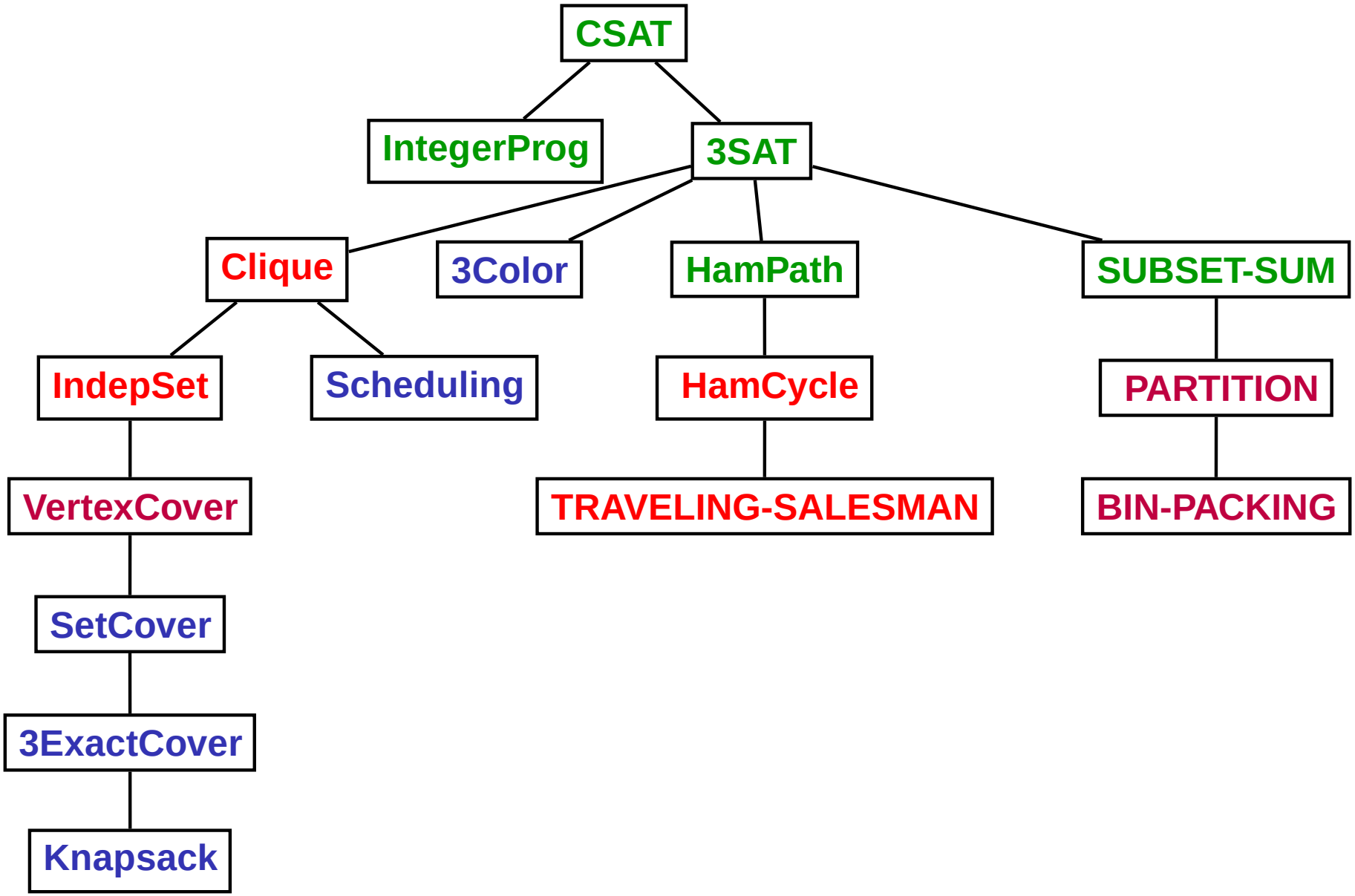
Section 4

Summary so Far

Summary So Far

- We have seen that SAT and CSAT are NP-complete.
- CSAT $\leq_P$ 3SAT
- 3SAT $\leq_P$ CLIQUE
- CLIQUE $\leq_P$ IND-SET
- In recitation: CLIQUE $\leq_P$ VC (Vertex Cover), IND-SET $\leq_P$ PARTITION, PARTITION $\leq_P$ BIN-PACKING
- Hence, all of the above languages are NP-complete
- Full list of NP-complete languages contains hundreds or thousands of known NP-complete problems (from combinatorics, operation research, VLSI design, computational geometry, bioinformatics, . . .).
- NP-completeness of new and of old problems is still established these days.

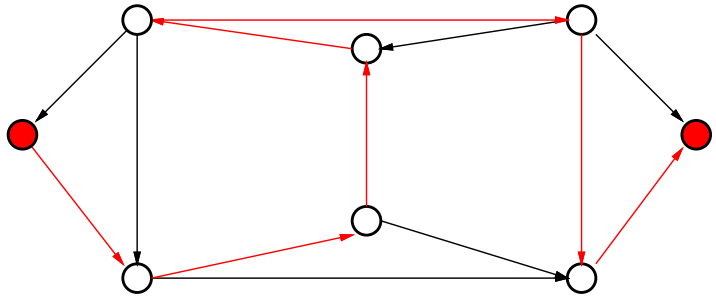
Chains of Reductions: NPC Problems (in red, seen so far, in green today.)



Section 5

Hamiltonian Path Is NP-Complete

(Directed) Hamiltonian Path



A Hamiltonian path in a directed G visits each node exactly once.

HAMPATH $= \{\langle G, s, t\rangle : G$ has Hamiltonian path from s to $t\}$

- We have seen that HAMPATH $\leq_P$ HAMCYCLE and HAMPATH $\leq_P$ TRAVELING-SALESMAN
- Will now show

Theorem 13

CSAT $\leq_P$ HAMPATH

thus establishing

Theorem 14

HAMPATH, HAMCYCLE, TRAVELING-SALESMAN $\in \mathcal{NPC}$

CSAT $\leq_P$ HAMPATH

For any CNF formula ϕ with clauses $c_1, \ldots, c_\ell$ and variables $x_1, \ldots, x_k$, we construct a directed graph G with vertices s and t, such that ϕ is satisfiable iff there is a directed Hamiltonian path from s to t.

Turn to a separate pdf presentation:

http://tau-cm.wdfiles.com/local--files/course-schedule/ham-reduction-new.pdf

Section 6

SUBSET-SUM Is NP-Complete

Reminder - SUBSET-SUM

$$\text{SUBSET-SUM} = \{\langle S = \{x_1, \ldots, x_k\}, t\rangle : \exists \{y_1, \ldots, y_m\} \subseteq S : \sum_{j=1}^{m} y_j = t\}$$

Example 15

- $\langle \{4, 11, 16, 21, 27\}, 25\rangle \in$ SUBSET-SUM because $4 + 21 = 25$.
- $\langle \{4, 11, 16, 21, 27\}, 26\rangle \notin$ SUBSET-SUM

- Will now show

Theorem 16

3SAT $\leq_P$ SUBSET-SUM.

thus establishing

Theorem 17

SUBSET-SUM $\in \mathcal{NPC}$.

3SAT $\leq_P$ SUBSET-SUM

Let ϕ be 3-CNF a boolean formula with

- variables $x_1, \ldots, x_k$
- clauses $C_1, \ldots, C_\ell$.

We “encode" ϕ into a set S containing $2k + 2\ell$ numbers in decimal notation, and a “target" number t, such that
$\phi \in$ 3SAT $\iff$ $(S, t) \in$ SUBSET-SUM

Variable Encoding

Each variable x_i is encoded as two $(k+\ell)$-digit numbers: y_i and z_i
Each decimal representation has two parts.

- Left-hand part, k digits: for both y_i and z_i, the i'th digit is 1, rest are 0s
- Right-hand part, ℓ digits: one digit for each clause
  - j^{th} digit of y_i is 1 if C_j contains x_i
  - j^{th} digit of z_i is 1 if C_j contains $\overline{x_i}$
  - rest are 0s

Example: $\phi = (x_1 \vee \overline{x_2} \vee \ldots) \wedge \ldots \wedge (\overline{x_1} \vee \overline{x_2} \vee \ldots)$

	1	2	...	k	C_1	...	C_ℓ
y_1	1	0	...	0	1	...	0
z_1	1	0	...	0	0	...	1
y_2	0	1	...	0	0	...	0
z_2	0	1	...	0	1	...	1

Clause Encoding

Each clause C_i is encoded into two equal $(k+\ell)$-digit numbers: g_i and h_i.
In both numbers, the $(k+i)$'th digit is 1 and rest are 0.

	1	...	k	C_1	C_2	...	C_ℓ
$\vdots$							
g_1	0	...	0	1	0	...	0
h_1	0	...	0	1	0	...	0
g_2	0	...	0	0	1	...	0
h_2	0	...	0	0	1	...	0

Target Encoding

The target t is assigned the following $(k + \ell)$-digit number

- first k digits are 1
- last ℓ digits are 3

	1	...	k	C_1	C_2	...	C_ℓ
$\vdots$							
t	1	...	1	3	3	...	3

Overall Encoding

	1	2	...	k	C_1	C_2	...	C_ℓ
y_1	1	0	...	0	1	0	...	0
z_1	1	0	...	0	0	0	...	1
y_2	0	1	...	0	0	0	...	0
z_2	0	1	...	0	1	0	...	1
$\vdots$								
g_1	0	0	...	0	1	0	...	0
h_1	0	0	...	0	1	0	...	0
g_2	0	0	...	0	0	1	...	0
h_2	0	0	...	0	0	1	...	0
$\vdots$								
t	1	1	...	1	3	3	...	3

$S = \{y_1, z_1, y_2, z_2, \ldots, y_k, z_k, g_1, h_1, g_2, h_2 \ldots, g_\ell, h_\ell\}$.

Transformation takes $O(n^2)$ time.

ϕ is satisfiable $\Longrightarrow$ $(S,t) \in$ SUBSET-SUM

Fix a satisfying assignment to ϕ, construct an initial solution V to (S,t) as follows: If $x_i = 1$ add y_i to V; otherwise, add z_i to V.

- Each of the first k digits of $\sum_{v \in V} v$ is 1 (as in t)
- For $1 \leq j \leq \ell$, consider the $(k+j)$'th digit of $\sum_{v \in V} v$.
  - at least 1 and not more than 3 literals in C_j are 1
    $\Longrightarrow$ the $(k+j)$'th digit is between 1 and 3.
  - Add "enough" $\{h_i, g_i\}$ to V to bring this digit up to 3.
    (Doing this does not effect the other digits of $(\sum_{v \in V} v)$).

Hence, $\sum_{v \in V} v = t$. ♣

$(S,t) \in$ SUBSET-SUM $\Longrightarrow$ ϕ is Satisfiable

Fix a solution V to (S,t).

1. Summation causes no carries
   - All digits in the numbers of S are either 0 or 1.
   - Since ϕ is in 3CNF, each "column" contains at most five 1s.
2. For each $i \in [k]$, the subset V contains one of y_i or z_i, but not both.
3. For each $i \in [\ell]$, the subset V contains at least one element of $\{y_1, z_1, \ldots, y_k, z_k\}$ whose $(k+i)$'th bit is 1:
   - Each of final ℓ "columns" sums to 3.
   - The set $\{g_i, h_i\}$ contributes at most 2
   - So at least one must comes from literal taking the value 1

Here is the satisfying assignment.

For $i \in [k]$: if $y_i \in V$ set $x_i = 1$; Otherwise set $x_i = 0$.

The assignment satisfies ϕ because

- No contradicting assignments (see Item 2)
- All clauses are satisfied (see Item 3)

♠

Section 7

Integer Programming (IP) is NP-Complete

Integer Programming (IP)

Definition 18

A linear inequality has the form

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$$

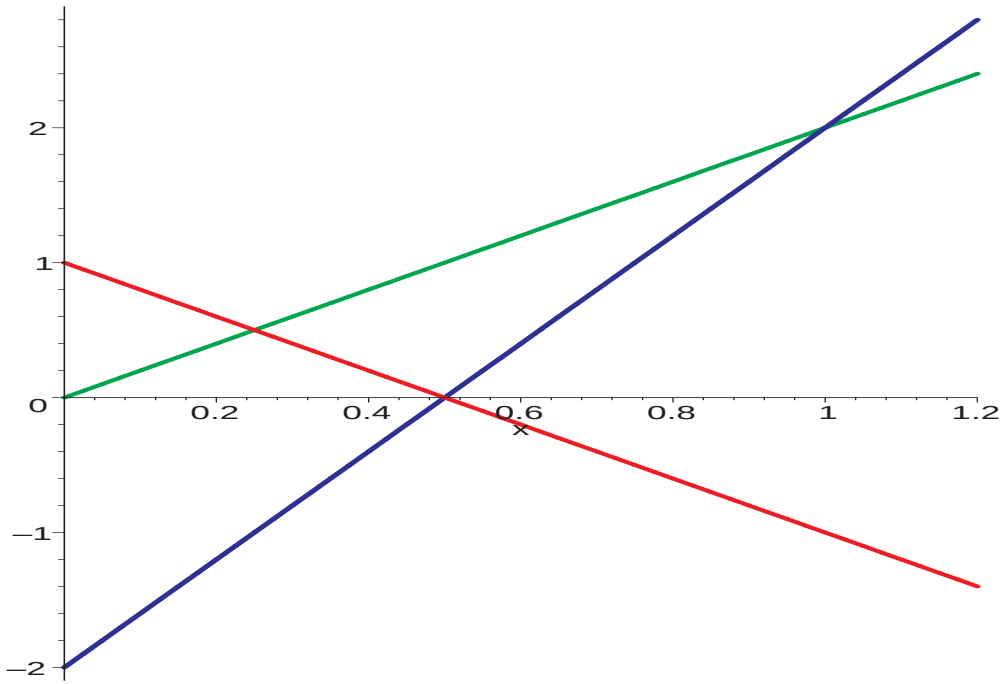
where $a_1, \ldots, a_n, b$ are real numbers, and $x_1, \ldots, x_n$ are real variables.

The Integer Programming (IP) problem]:
Input: A set of linear inequalities with integer coefficients in variables $x_1, x_2, \ldots, x_n$.
Question: Does this set has an integer solution – all x_i are integers?

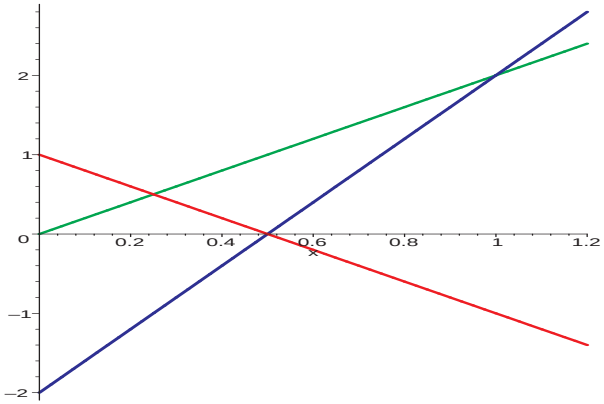
Integer Programming: Example

Consider the following system of linear inequalities

$$
\begin{aligned}
y &\leq 2x && \text{below green line} \\
-2x + 1 &\leq y && \text{above red line} \\
4x - 2 &\leq y && \text{above purple line} \\
0 \leq x &\leq 1 \\
0 \leq y &\leq 2
\end{aligned}
$$



Integer Programming: Example, 2



This set does have a unique (integer) solution: the right hand corner of the solid triangle, $(1, 2)$.

But if we change the constraint on y to $0 \leq y \leq 1$, then we'd have no solution with integer coordinates, even though there are many solutions with rational, or real, coordinates.

The Language IP

The language of integer linear inequalities with integer solution:

$$\text{IP} = \{\langle e_1, \ldots, e_m \rangle : e_1, \ldots, e_m \text{ are integer linear inequalities with (joint) integer solution}\}$$

Theorem 19

$\text{IP} \in \mathcal{NPC}$.

IP $\in \mathcal{NP}$

Question 20

Do we always have a small enough certificate?

Consider the following equations

$$\begin{aligned} x_0 &= 1 \\ x_1 &= 2x_0 \\ &\vdots \\ x_m &= 2x_{m-1} \end{aligned}$$

The solution $x_m = 2^m$!

Question 21

Solving a linear set of m equations with m unknowns: $Ax = b$, where $|a_{i,j}|, |b_i| < M$. How large can $|x_i|$ get?

Answer: Recall Cramer's Rule, using determinants. $|A| < m! \cdot M^m$.
(Same holds for inequalities.)
Hence, solution size is polynomial bounded.

CSAT $\leq_P$ IP

$$\text{CSAT} = \{\langle\phi\rangle : \ \phi \text{ is a satisfiable CNF Boolean formula}\}$$

Let ϕ be a CNF formula with ℓ clauses and k variables $x_1, \ldots, x_k$.

We reduce ϕ to an IP instance E with $2k$ variables $x_1, y_1, \ldots, x_k, y_k$, $\ell + 2k$ linear inequalities, and k linear equalities.

- Each x_i in ϕ corresponds to the variable x_i in E.
- Each $\overline{x_i}$ in ϕ corresponds to the variable y_i in E.
- For each i, we add E the inequalities $x_i \geq 0$, $y_i \geq 0$, and the equality $x_i + y_i = 1$ (what do these three express?)
- For each clause C of ϕ, we add E the inequality $\sum_{z \in C} z \geq 1$ (what does this inequality express?)
  For example, $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$ is translated to $x_1 + y_2 + y_3 + x_4 \geq 1$.

Reduction Example

$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$ is translates to

$$
\begin{aligned}
x_1 + y_2 + y_3 + x_4 &\geq 1 \\
x_3 + y_5 + x_6 &\geq 1 \\
x_3 + y_6 &\geq 1 \\
x_1 \geq 0,\ y_1 \geq 0,\ x_1 + y_1 &= 1 \\
x_2 \geq 0,\ y_2 \geq 0,\ x_2 + y_2 &= 1 \\
x_3 \geq 0,\ y_3 \geq 0,\ x_3 + y_3 &= 1 \\
x_4 \geq 0,\ y_4 \geq 0,\ x_4 + y_4 &= 1 \\
x_5 \geq 0,\ y_5 \geq 0,\ x_5 + y_5 &= 1 \\
x_6 \geq 0,\ y_6 \geq 0,\ x_6 + y_6 &= 1
\end{aligned}
$$

CSAT $\leq_P$ IP: Validity (sketch)

Should show

1. Reduction, g, is poly-time computable
2. $\phi \in \text{CSAT} \implies g(\phi) \in \text{IP}$
3. $g(\phi) \in \text{IP} \implies \phi \in \text{CSAT}$.

1. Poly time: easy (verify details!).
2. Suppose $\phi \in \text{CSAT}$. Take a satisfying assignment.
   If $x_i = 1$, in $g(\phi)$ assign $x_i = 1, y_i = 0$.
   If $x_i = 0$, in $g(\phi)$ assign $x_i = 0, y_i = 1$.
   - "Sanity check" constraints (i.e., $x_i \geq 0$, $y_i \geq 0$, $x_i + y_i = 1$) are satisfied.
   - "Clause constraints" (i.e., $\sum_{z \in C} z \geq 1$) are satisfied due to at least one literal satisfied in each clause.
   - Implying $g(\phi) \in \text{IP}$.
3. $g(\phi) \in \text{IP} \implies \phi \in \text{CSAT}$, is similar.