Part V: other problems

Satisfiability (SAT)

Definition: A Boolean variable takes a value either false (F) or true (T).

Definition: A Boolean formula takes any of the following forms:

- g : true if g is true
- $\neg g$: "not g" : true if g is false
- $g \vee h$: "g or h" : true if g or h is true (or both)

- $g \wedge h$: "g and h": true if both g and h are true.

where g, h are Boolean variables or other Boolean formulas.

Example $x \vee (\neg y \wedge z)$

Notation: $\bigwedge_{i=1,\dots,n} x_i = x_1 \wedge x_2 \wedge \dots \wedge x_n$

$$\bigvee_{i=1,\dots,n} x_i = x_1 \vee x_2 \vee \dots \vee x_n$$

Definition An assignment for a set of variables gives a value (T or F) to each variable.

Example $x=T$, $y=F$, $z=T$

Definition A Boolean formula is satisfiable if there exists an assignment for its variables such that the value of the formula is true.

Otherwise, it is unsatisfiable.

Example $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$
$\wedge (x \vee \neg y)$

is **SAT**. **proof**: Let $x = T$, $y = F$, $z = T$

Definition: **SAT problem**: Given a Boolean formula, find an assignment, or prove that it is UNSAT.

Example: We organize a wedding dinner, with invitees $N = \{1, \dots, n\}$ and tables $K = \{1, \dots, k\}$. Each invitee hates a set of other invitees $H_i \subseteq N$, for all i. Find a seating arrangement such that noone is seated with someone they hate.
Tables can seat any number of invitees, and invitees can be assigned multiple tables.

VAR:

$$x_{ij} = \begin{cases} \underline{\text{true}} & \text{if invitee } i \text{ is at table } j \\ \underline{\text{false}} & \text{otherwise,} \end{cases}$$

for $i \in N$, $j \in K$.

MODEL:

$$\bigwedge_{i \in N} \Big(\bigvee_{j \in K} x_{ij} \Big)$$

each invitee assigned at least one table

$$\wedge \bigwedge_{i \in N} \bigwedge_{j \in K} \Big(\neg x_{ij} \vee \big(\bigwedge_{\ell \in H_i} \neg x_{\ell j} \big) \Big)$$

either i is not at table j or none of H_i is at table j.

What would an IP model look like?

<u>VAR</u>: $x_{ij} = \begin{cases} 1 & \text{if } i \text{ is at table } j \\ 0 & \text{otherwise} \end{cases}$

<u>MODEL</u>:

$$\begin{aligned} \min\ & 0 \\ \text{s.t.}\ & \sum_{j \in K} x_{ij} \geq 1, \quad \forall i \in N \\ & \sum_{\ell \in M_i} x_{\ell j} \leq (1 - x_{ij}) \cdot |M_i|, \quad \forall i \in N,\ \forall j \in K \\ & x_{ij} \in \{0, 1\}, \quad \forall i \in N, \forall j \in K \end{aligned}$$

Boolean reformulations

1) $\wedge$ and $\vee$ are commutative and associative

2) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

3) $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

4) $x \wedge (x \vee y) = x$

5) $x \vee (x \wedge y) = x$

6) $\neg(x \vee y) = \neg x \wedge \neg y$

7) $\neg(x \wedge y) = \neg x \vee \neg y$

<u>proof:</u> 7)

x	y	$\neg(x \wedge y)$	$\neg x \vee \neg y$
F	F	T	T
F	T	T	T
T	F	T	T
T	T	F	F

<u>rest: exercise</u>

Definition A *literal* is a Boolean variable x or its negation $\neg x$.

Definition A *clause* is an OR of literals:

$$\bigvee_{j \in C^+} x_j \vee \bigvee_{j \in C^-} \neg x_j$$

Definition A formula is in conjunctive normal form (CNF) if it is an AND of clauses:

$$\bigwedge_{i=1,\dots,m} \left(\bigvee_{j \in C_i^+} x_j \vee \bigvee_{j \in C_i^-} x_j \right)$$

Example

$$\underbrace{(\underbrace{x \vee \neg y \vee z}_{\text{clause}}) \wedge (\underbrace{x \vee y \vee \neg z}_{\text{clause}}) \wedge (\underbrace{\neg x \vee y \vee z}_{\text{clause}})}$$

is in CNF.

Theorem Every Boolean formula can be put in CNF, whose size is polynomial in the size of the original formula (if we allow additional variables).

Remark:

- The Disjunctive Normal Form (DNF) is an OR of ANDs of literals.
- The size of the DNF can be exponential.
- The DNF is simply a list of the assignments that satisfy the formula.
- example:

$$(x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (y \wedge z)$$

assignments:

1) $x = T$, $y = F$, $z = F$
2) $x = F$, $y = F$, $z = T$
3) $x = F$ or T, $y = T$, $z = T$

How do we solve SATs

Consider a formula in CNF with variables x_i, $i = 1, \ldots, n$.

<u>Naive method</u>: enumerate all 2^n possible assignments. Check the formula for each.

<u>Better method</u>: Backtracking

Set $x_1 = F$

if some clause is just (x_1), UNSAT

otherwise, simplify formula, solve it.

Set $x_1 = T$

if some clause is just $(\neg x_1)$, UNSAT

otherwise, simplify formula, solve it.

Example: $(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2)$

$x_1 = F$ / $x_1 = T$

$(T \vee \neg x_2 \vee x_3) \wedge (F \vee x_2)$

$= T \wedge (x_2)$

$= x_2$

$x_2 = F$ / $x_2 = T$

$F \to$ UNSAT

$T \to$ SAT

Assignment:

$x_1 = F$

$x_2 = T$

$x_3 = T$ or F

Any SAT in CNF can be formulated as an IP

$$\bigwedge_{i=1,\dots,m} \left(\bigvee_{j\in C_i^+} x_j \vee \bigvee_{j\in C_i^-} \neg x_j \right)$$

$$\Leftrightarrow \quad \begin{aligned} \min\ & 0 \\ \text{s.t. } & \sum_{j\in C_i^+} x_j + \sum_{j\in C_i^-} (1-x_j) \geq 1, \quad \forall i=1,\dots,m \\ & x_j \in \{0,1\}, \quad \forall j=1,\dots,n \end{aligned}$$

Correct in theory.

In practice, NEVER do that.

Why:

- no objective function $\Rightarrow$ no pruning
- consider a node of b&b / backtracking tree

If one constraint / clause has just one variable, we should just fix it:

$$x_j \geq 1 \quad \Rightarrow \quad x_j = 1$$

$$1 - x_j \geq 1 \quad \Rightarrow \quad x_j = 0$$

Otherwise: CLAIM: the LP relaxation is always feasible.

<u>proof</u> Set $x_j = \frac{1}{2}$, $\forall j$.

$\Rightarrow$ the LP relaxation tells us <u>nothing</u>