1) Class : Discussion on TSP + OptMap.
2) SAT.
3) QUIZ
4) Architecture Team for SAT.

BOOLEAN SATISFIABILITY

<u>Logic</u> : a) Philosophical Logic:
Aristotle (STOIC) : Philosophy <u>vs</u> Sophistry.

b) Constructivist :
Leibnitz (1646 - 1716) - Lingua Characteristica Universalis.
George Boole (1815 - 1864) - Making Logic Algebraic.
Jevons (1835 - 1882) - Logic Piano (Making Logic Computational)

Arithmetization $(T, F \Rightarrow 0,1)$
$\Rightarrow$ Axiomatization $(A \to B \to \neg B \to \neg A)$
$\Rightarrow$ Algebraization
$\Rightarrow$ Algorithmization.

⋮

Propositional Logic (PL).

Boolean Connectives: Negation $\neg$
Conjunction $\wedge$
Disjunction $\vee$

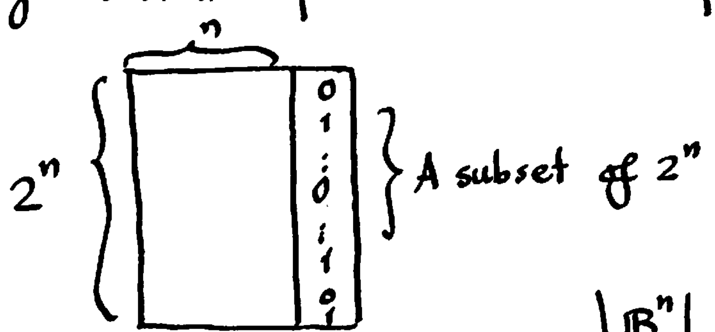
$\neg : \{0,1\} \to \{0,1\}$
$: 1 \mapsto 0 \; ; \; : 0 \mapsto 1$

$\wedge : \{0,1\}^2 \to \{0,1\}$
$: (1,1) \mapsto 1; \quad (1,0) \mapsto 0;$
$: (0,1) \mapsto 0; \quad (0,0) \mapsto 0$

$\vee : \{0,1\}^2 \to \{0,1\}$
$: (1,1) \mapsto 1; \quad (1,0) \mapsto 1$
$: (0,1) \mapsto 1; \quad (0,0) \mapsto 0.$

Boolean Truth Function:

A function $f : \{0,1\}^n \to \{0,1\}$ is called an n-ary Boolean function or truth function.



$$|\mathbb{B}^n| = 2^{2^n}$$

A Boolean function is satisfiable $\models f$

iff $\exists_{x \in \{0,1\}^n} \; f(x) = 1.$ $\leftarrow \in NP$

Also NP-complete.

Propositional Language:

Boolean formulas:

$\mathcal{F}$ of formulas built up from the symbols

$($, $)$, $\wedge$, $\vee$, $\neg$ ... and

logical variables $x_1, x_2, \cdots, x_n$, inductively as follows:

(F_1) The atomic strings $x_1, x_2, \cdots$ are formulas, called prime formulas (also atomic formulas).

(F_2) If the strings α and β are formulas, then so too are strings

$$(\alpha \wedge \beta), \quad (\alpha \vee \beta) \text{ and } \neg\alpha.$$

Truth assignment:

$$\omega : PV \to \{0,1\}$$

Extend to $\omega : \mathcal{F} \to \{0,1\}$

$$\omega \neg \alpha \equiv 1 - \omega\alpha$$
$$\omega(\alpha \wedge \beta) \equiv \omega\alpha \cdot \omega\beta$$
$$\omega(\alpha \vee \beta) \equiv \max(\omega\alpha, \omega\beta)$$

<u>THM</u> Every Boolean function can be represented by a Boolean formula $\in \mathcal{F}$.

$\square$.

NORMAL FORMS:

1) Literals: Prime formulas and negation of prime formulas are called literals

$$x_i \;,\; \neg x_i \; (\equiv \bar{x}_i)$$

2) Conjunctive Normal Form (CNF)

A conjunction

$$\beta_1 \wedge \beta_2 \wedge \cdots \wedge \beta_m$$

where each β_i is a disjunction of literals, is called a Conjunctive Normal form:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee x_4 \vee \neg x_5) \wedge \cdots \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Variable Clause literal

$$\langle lit \rangle := x_i \mid \neg x_i$$
$$\langle clause \rangle := \langle lit \rangle \vee \langle lit \rangle \vee \langle lit \rangle \vee \cdots \vee \langle lit \rangle$$
$$\langle cnf \rangle := \langle clause \rangle \wedge \langle clause \rangle \cdots \wedge \langle clause \rangle .$$

3) Disjunctive Normal Form (DNF)

A disjunction

$$\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_\ell$$

where each α_i is a conjunction of literals, is called a Disjunctive Normal Form.

Let $\omega : PV \to \{0,1\}$ = Truth Assignment

α = Boolean Formula in CNF (K-CNF)

$$\exists \omega \;\; \omega \models \alpha \quad \equiv \quad \exists \omega \;\; \omega\alpha = 1$$

= Boolean Satisfiability Problem

SAT, K-SAT (3-SAT & 2-SAT)

Logic Problems.

SAT — Is a formula satisfiable? — NP-complete.

$$\exists \omega \quad \omega \models \alpha$$

(α = CNF)

TAUT — Is a formula a tautology? — coNP-complete

$$\forall \omega \quad \omega \models \alpha \qquad \left(\exists \omega \quad \omega \models \neg\alpha\right)$$

Negation of

(α = CNF)

EQUIV — Are two formulas semantically equivalent? — coNP-complete

$$\forall \omega \quad \omega \models \alpha \leftrightarrow \beta$$

(α = CNF
$\beta = x_i \vee \neg x_i$)

1) K-SAT $\rightsquigarrow$ 3-SAT. (3-SAT = NPC).

Clauses: $C = \{c_1, c_2, \dots, c_m\}$

Variables: $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$

$c_i \in C \;\rightarrow\;$ literals $\Rightarrow \{z_1, z_2, \dots, z_K\}$ $\qquad z_j \in \{u_{jr}, \bar{u}_{jr}\}$

For each clause c_i introduce additional variables:

$$\{y_{i_1}, y_{i_2}, \dots, u_{i,k-3}\} \qquad K \geq 3$$

$$\{y_{i_1}, y_{i_2}\} \qquad K \leq 3$$

$k=1 \qquad C_i = z_1 \qquad C_i' = (z_1 \vee y_{i_1} \vee y_{i_2})$
$$\wedge (z_1 \vee \bar{y}_{i_1} \vee y_{i_2})$$
$$\wedge (z_1 \vee y_{i_1} \vee \bar{y}_{i_2})$$
$$\wedge (z_1 \vee \bar{y}_{i_1} \vee \bar{y}_{i_2})$$

$k=2 \qquad C_i = z_1 \vee z_2 \qquad C_i' = (z_1 \vee z_2 \vee y_{i_1})$
$$\wedge (z_1 \vee z_2 \vee \bar{y}_{i_1})$$

$k=3 \qquad C_i = z_1 \vee z_2 \vee z_3 \qquad C_i' = (z_1 \vee z_2 \vee z_3)$

$k>3 \qquad C_i = (z_1 \vee z_2 \vee \cdots \vee z_k)$

$$C_i' = (z_1 \vee z_2 \vee y_{i_1}) \wedge (\bar{y}_{i_1} \vee z_3 \vee y_{i_2}) \wedge \cdots$$
$$\wedge (\bar{y}_{i,l-2} \vee z_l \vee y_{i,l-1}) \wedge \cdots$$
$$\wedge (\bar{y}_{i,k-3} \vee z_{k-1} \vee z_k)$$

a) $z_1 = T$ or $z_2 = T \qquad \forall_j \; y_{ij} = F.$

b) $z_{k-1} = T$ or $z_k = T \qquad \forall_j \; y_{ij} = T$

c) $z_l = T \qquad \forall_{j \le l-2} \; y_{ij} = T \quad \& \quad \forall_{j \ge l-1} \; y_{ij} = F$

d) $\forall_l \; z_l = F \qquad C_i' \equiv y_{i_1} \wedge (\bar{y}_{i_1} \vee y_{i_2}) \wedge \cdots \wedge (\bar{y}_{i,k-4} \vee y_{i,k-3}) \wedge \bar{y}_{i,k-3}$

$$\equiv y_{i_1} \wedge (y_{i_1} \to y_{i_2}) \wedge \cdots (y_{i,k-4} \to y_{i,k-3}) \wedge \bar{y}_{i,k-3}$$

$$\equiv y_{i,k-3} \wedge \bar{y}_{i,k-3} \equiv \bot$$

3-SAT = NP Complete.

Two Ideas (Polynomial Cases)

KROM-SAT

KROM-Clause

{ Every clause is a disjunction of ≤ 2 literals. }

$$x_1 \vee \neg x_3$$
$$\equiv \neg x_1 \to \neg x_3$$
$$x_1$$
$$\equiv T \to x_1$$
$$\vdots$$

Strongly Connected Component in a graph $G = \langle V, E \rangle$

$$V = \{ x_i, \neg x_i \mid 1 \leq i \leq n \}$$

$(u, v) \in E$ iff $u \to v$
$\equiv \neg v \to \neg u$
iff $(\neg v, \neg u) \in E$

HORN-SAT.

HORN-Clause

{ Every clause is a disjunction of literals in which all or nearly all of the literals are negated }

$$x \vee \neg y \vee \neg z$$
$$\equiv y \wedge z \to x$$
$$\neg x \vee \neg y$$
$$\equiv x \wedge y \to \bot$$
$$x$$
$$\equiv T \to x$$

Truth Propagation.

Initialize : ~~[illegible]~~ $\omega : PV \to 0$

satisfies { $x \wedge y \to \bot$, $x \wedge y \wedge z \to v$ }

But not $T \to x$ (unit clause)

Update : Find a clause whose r.h.s is not satisfied FLIP it

Repeat : ...

TWO RANDOMIZED ALGORITHMS. 3-SAT.

1) <u>while</u> # iterations $< 20\left(\frac{3}{2}\right)^m$ <u>do</u> # clauses.

Initialize $G := F$

<u>while</u> $G \not\equiv$ 2-CNF <u>do</u>

Choose a 3-CNF clause in G, $\equiv c$

$c \to c'$ (remove a literal from c randomly)

$G := (G \setminus c) \cup \{c'\}$

Run 2-SAT on G

<u>If</u> $G =$ SAT ~~not~~ <u>then</u> return TRUE ~~else~~

return FALSE.

2) <u>while</u> # iterations $< 10n^2\left(\frac{4}{3}\right)^n$ <u>do</u>

Let A be an assignment for F (Random)

<u>while</u> # iterations $< 3n$ <u>do</u>

<u>if</u> A satisfies F, return SAT.

<u>else</u> find a clause $c \in F$ not satisfied

Randomly pick a literal and FLIP.

Best complexity bound is $(1.308)^n$ <u>Hertli</u>.