MaxSAT: Maximum Satisfiability

Logic in Computer Science

1

3SAT vs 2SAT

- SAT: Decide if a set of clauses is satisfiable.
- 3SAT: Decide if a 3CNF is satisfiable.
- 2CNF: Each clause contains at most 2 literals.
- 2SAT: Decide if a 2CNF is satisfiable.

Theorem: 2SAT can be solved in polynomial time.

2

2SAT is in P

Theorem: 2SAT is polynomial-time decidable.

Proof: We'll show how to solve this problem efficiently using path searches in graphs…

3

3

Searching in Graphs

Theorem: Given a graph $G=(V,E)$ and two vertices $s,t \in V$, finding if there is a path from s to t in G takes $O(|V|+|E|)$ time.

Proof: Use some search algorithm (DFS/BFS). ■

4

4

Graph Construction

G = (V, E)

- V: Vertex for each variable and the negation of a variable
- E: Edge (α, β) iff there exists a clause equivalent to $(\neg\alpha \mid \beta)$.
- Every clause $(A \mid B)$ of 2CNF contributes two edges: $(\neg A, B)$ and $(\neg B, A)$.

5

5

Observation

Claim: If the graph contains a path from α to β, denoted by $\alpha \Rightarrow \beta$, it also contains a path from $\neg\beta$ to $\neg\alpha$.

Proof: If there's an edge (α, β), then there's also an edge $(\neg\beta,\neg\alpha)$. Both edges come from clause $(\neg\alpha \mid \beta)$ and $\alpha \rightarrow \beta \equiv \neg\beta \rightarrow \neg\alpha$.

6

6

Correctness

Thereom: A 2CNF formula φ is unsatisfiable iff there exists a variable x, such that:

1. there is a path from x to $\neg x$ in the graph
2. there is a path from $\neg x$ to x in the graph

That is, x and $\neg x$ are in a cycle.

7

7

Correctness (1)

- Suppose there are paths $x \Rightarrow \neg x$ and $\neg x \Rightarrow x$ for some variable x, then both $x \rightarrow \neg x$ and $\neg x \rightarrow x$ are true, because the implication relation is transitive. However,
- $(x \rightarrow \neg x) \wedge (\neg x \rightarrow x) \equiv (x \leftrightarrow \neg x) \equiv$ false

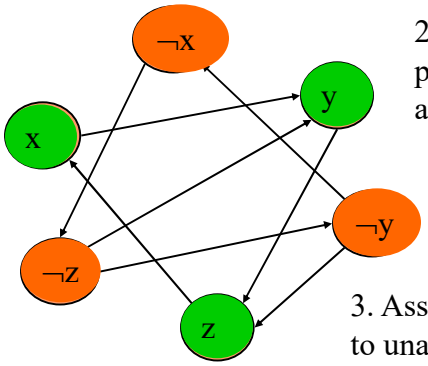
8

8

Correctness (2)

- S = { (-x | y), (x | -z), (y | z), (-y | z) }
- Suppose v and ¬v are not in a cycle for any v.
- Construct an assignment as follows:

1. If there is a path ¬x ⇒ x, assign x = 1;
   If there is a path x ⇒ ¬ x, assign x = 0;



2. If there is a path x ⇒ y, x = 1, assign y = 1.

3. Assign arbitrary values to unassigned vertices

9

9

Correctness (2)

Claim: The algorithm of assigning truth values is well defined.

Proof: The path relation x ⇒ y represents the implication x → y. If x = 1, then it must be the case that y = 1, so that x → y is true.

If there exists a path ¬x ⇒ x ⇒ y ⇒ ¬y, such that x = 1 and y = 0 by the first line of the algorithm, but x ⇒ y is a violation of the second line, then from x ⇒ y, we have ¬y ⇒ ¬x, thus, ¬x, x, y, ¬y are in a cycle, a contradiction to the condition that ¬x, x, are not in a cycle.

10

10

2SAT is in P

We get the following efficient algorithm for 2SAT:

- For each variable x find if there is a path from x to ¬x and vice-versa.
- Reject if any of these tests succeeded.
- Accept otherwise.

⇒ 2SAT∈P. ■

11

11

2SAT can done in linear time

- Use SCC (Strongly Connected Components) to group vertices which share a common cycle.
- x and ¬x are in a cycle iff they are in the same SCC.
- SCC can be found in the time O(|V|+|E|)
- https://en.wikipedia.org/wiki/Strongly_connected_component

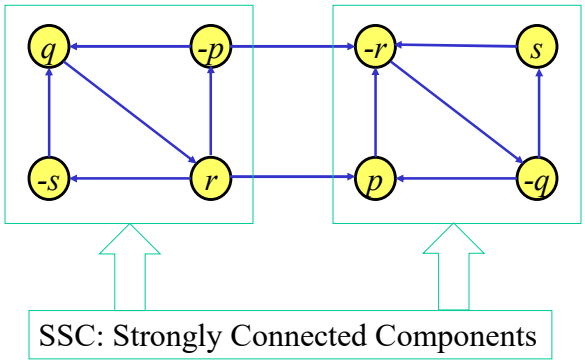
12

12

2SAT is linear solvable

$C = \{ c_1: (p \mid q),\ c_2: (-p \mid -r), c_3: (p \mid -r),$
$c_4: (-q \mid r), c_5: (q \mid s),\ c_6: (-r \mid -s) \}$

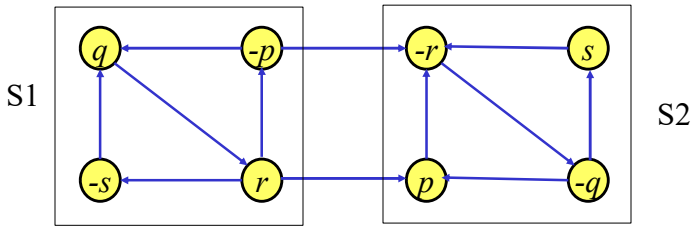
Implication graph of C:



13

2SAT is linear solvable

$C = \{ c_1: (p \mid q),\ c_2: (-p \mid -r), c_3: (p \mid -r),$
$c_4: (-q \mid r), c_5: (q \mid s),\ c_6: (-r \mid -s) \}$



SCC: S1 = { q, r, -s, -p }, S2 = { p, -q, -r, s }
G' = ({ S1, S2}, { (S1, S2) }
G' has no cycles; the topological order is S1, S2.
Assign 0 to S1, 1 to S2.

14

What is MaxSAT?

The problem of determining the maximum number of clauses, of a given propositional formula in conjunctive normal form (CNF), that can be made true by an interpretation of the formula.

What is the MaxSAT solution for S?
S = { (¬p), (p | q), (p | ¬q), (p | r), (p | ¬ r), (q | r) }
Solutions: σ = { p, q, r }, { p, q, ¬ r }, or { p, ¬ q, r }.
Five satisfied and one falsified.

15

Why Bother?

- Simply determining that an instance is UNSAT may not be enough.
- We want the optimal way to make the instance satisfiable by allowing for *some* clauses to be unsatisfied.
- AI
  - University course scheduling
  - …
- EDA (Electronic Design Automation)
  - Over constrained system analysis
  - FPGA routing
  - …

16

SAT: Propositional Satisfiability

SAT: Decide if a formula (CNF) is satisfiable

$$
\begin{array}{rcl}
a & \wedge & (\textbf{C1}) \\
(\neg a \vee b) & \wedge & (\textbf{C2}) \\
(\neg b \vee c) & \wedge & (\textbf{C3}) \\
(\neg c \vee d) & \wedge & (\textbf{C4}) \\
\neg d & & (\textbf{C5})
\end{array}
$$

17

17

Maximum Satisfiability (Max-SAT)

MaxSAT: Find an interpretation to maximize the number of satisfiable clauses, or to minimize the number of falsified clauses.

$$
\begin{array}{rcl}
a & \wedge & (\textbf{C1}) \\
(\neg a \vee b) & \wedge & (\textbf{C2}) \\
(\neg b \vee c) & \wedge & (\textbf{C3}) \\
(\neg c \vee d) & \wedge & (\textbf{C4}) \\
\neg d & & (\textbf{C5})
\end{array}
$$

Solution: a = 1, b = 1, c = 1, d = 1 **C5 is false**

18

18

Weighted MaxSAT

Weighted MaxSAT: Assign weights to clauses and find an interpretation to maximize an objective.

20:	a ∧	(C1)
30:	$\neg a \vee b$ ∧	(C2)
4:	$\neg b \vee c$ ∧	(C3)
2:	$\neg c \vee d$ ∧	(C4)
7:	$\neg d$	(C5)

=

Maximize
20×C1+30×C2+ 4×C3+2×C4+7×C5
for those satisfied clauses.
Or **minimize** the cost for those falsified clauses.

Solution: a = 1, b = 1, c = 1, d = 0
C4 is false and Objective = 61 (or cost = 2).

19

19

Hybrid MaxSAT

Hybrid MaxSAT: Clauses are divided into hard/soft subsets; assign weights to soft clauses and find an interpretation to maximize an objective and satisfy all hard clauses.

	a ∧	(C1)
	$\neg a \vee b$ ∧	(C2)
4:	$\neg b \vee c$ ∧	(C3)
2:	$\neg c \vee d$ ∧	(C4)
7:	$\neg d$	(C5)

=

Subject to	**C1 and C2**
Maximize	**4×C3+2×C4+7×C5**
or Minimize	**4×$\overline{\text{C3}}$+2×$\overline{\text{C4}}$+7×$\overline{\text{C5}}$**

Solution: a = 1, b = 1, c = 1, d = 0
C4 is false and Objective = 11 (cost = 2)

20

20

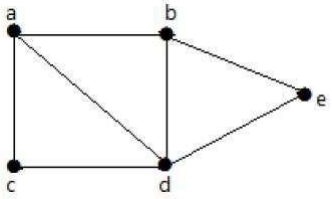
Maximum Clique of a Graph

- Encode the maximum clique problem in MaxSAT
- Propositional variables (True means in the clique):
  { *a, b, c, d, e* }

Hybrid MaxSAT:

Hard = { (-*a*| -*e*), (-*b* | -*c*), (-*c* | -*e*) }

Soft = { (*a*; 1), (*b*; 1), (*c*; 1), (*d*; 1), (*e*; 1)}



Weighted MaxSAT:

C = { (-*a*| -*e*; 5), (-*b* | -*c*; 5), (-*c* | -*e*; 5), (*a*; 1), (*b*; 1), (*c*; 1), (*d*; 1), (*e*; 1)}

21

Weighted vs Hybrid: Same Power

Increased Expressive Power for Applications

Hard — **Soundness Conditions**

+

Soft — **Objectives**

- **Balancing tradeoffs** (e.g., precision vs. scalability)
- **Handling uncertainty** (e.g., incorrect specs)
- **Modeling missing data** (e.g., partial programs)
- …

22

22

Applications of MaxSAT

Many real-world applications can be easily and neatly encoded to MaxSAT:

- Eclipse platform uses MaxSAT for managing the plugins dependencies
- Error localization in C code
- Debugging of hardware designs
- Reasoning over Biological Networks (Genes)
- Course timetabling
- Final exam scheduling
- . . .

23

Applications of MaxSAT in Automotive (Re)Configuration

Applications of MaxSAT in Automotive Configuration, by: Rouven Walter and Christoph Zengler and Wolfgang Kuchlin

The after-sales business asks for extensions, replacements, or removal of components of a valid configuration with minimal effort.
Automotive configuration can be represented as a CNF formula in propositional logic, where each model is called a valid configuration of a car.
http://ceur-ws.org/Vol-1128/paper3.pdf

24

MaxSAT

- Very hard problems --- NP-HARD
- Optimization of SAT
- Difficult to find an approximate solution of the problem
- Large-size MAX-SAT instances cannot be solved exactly, and one must resort to approximation algorithms and heuristics.

25

Max2SAT

- Instance: A 2CNF formula φ and a goal K.
- Problem: To decide if there is an assignment satisfying at least K of φ's clauses.

Example: a 2CNF formula

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

26

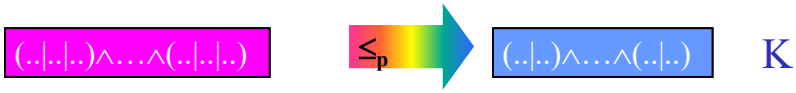
26

Max2SAT is NP Complete

Theorem: Max2SAT is NP-Complete.

Proof: Max2SAT is clearly in NP.

We'll show 3SAT $\leq_p$ Max2SAT.



27

27

Gadgets

Claim: Let

$\psi(x,y,z,w) =$ $(x)\wedge(y)\wedge(z)\wedge(w)\wedge$
$(\neg x|\neg y)\wedge(\neg y|\neg z)\wedge(\neg z|\neg x)\wedge$
$(x|\neg w)\wedge(y|\neg w)\wedge(z|\neg w).$

- Every satisfying assignment for $(x|y|z)$ can be extended into an assignment which satisfies exactly 7 of the clauses.
- Other assignments can satisfy at most 6 of the clauses.

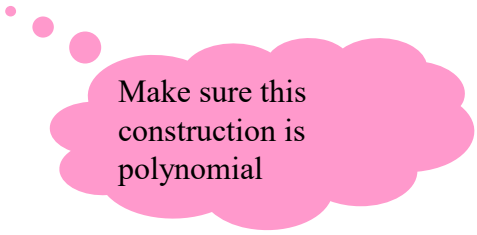
Proof: By truth table on vars.

28

28

The Construction

- For each $1 \le i \le m$ of clauses, replace the i-th clause of the 3CNF formula $(\alpha|\beta|\gamma)$ with a corresponding $\psi(\alpha,\beta,\gamma,x_i)$ to get a 2CNF formula.
- Take $K=7m$.



29

29

Correctness

- Every satisfying assignment for the 3CNF formula can be extended into an assignment which satisfies $7m$ clauses in 2CNF.
- If $7m$ clauses of the 2CNF formula are satisfied, each clause ψ has 7 satisfied clauses, so the original formula is satisfied.

30

30

Conclusion

- $3SAT \leq_p Max2SAT$ and $Max2SAT \in NP$
- Max2SAT is NP-Complete. ■

31

31

2SAT vs 3SAT vs Max2SAT

- We've seen that checking if a given CNF formula is satisfiable:
  - Polynomial-time decidable, if every clause contains up to 2 literals (2SAT).
  - NP-hard, if each clause has up to 3 literals (3SAT).
  - NP-hard, if each clause contains up to 2 literals and we look for an optimal solution (Max2SAT).

32

32

Solving MaxSAT

- Traditional approaches for MaxSAT:
  - Systematic search: check every interpretation to see which interpretation is optimal. It's complete but has limited scalability
  - Local search: randomly start with an interpretation and search its neighborhood for a better one. It's scalable but sometimes far from optimum (local optimum)

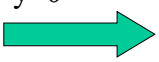
33

33

Assignments from case-split

{(x|y|z; 3),
(¬x|y; 2)}

(y;2)



(□;2)

□ empty clause.
It cannot be satisfied,
2 is necessary cost

We may use case-splits as in DPLL to try all possible interpretations, to find which one is optimal. – This is a systematic search.

34

34

Systematic Search: 2 Key Features

The DPLL process, and extensions, for SAT

- Search guided by
  - "Well-informed" heuristics
  - Clauses falsified or made "critical" during search
- Efficiency relies heavily on the ability to avoid local inconsistencies and prune the search space
  - **Unit propagation**:
    if a=0, b=0, and have clause (a or b or ¬c),
    better force c=1 right away
  - **Clause learning**:
    if a search branch reaches a contradiction,
    learn *why* this happened and never let this happen again

35

35

Systematic Search: 2 Key Features

Unfortunately, unit propagation and clause learning are not suitable for case-split MaxSAT solvers.

- Unit resolution is not sound for MaxSAT
  - S = { (¬p), (p | q), (p | ¬q), (p | r), (p | ¬ r), (q | r) }

  Using unit resolution, S is simplified to S':
  - S' = { (¬p), (q), (¬q), (r), (¬ r), (q | r) }

  S's minimal cost is 1; the cost of S' is at least 2
- How to assign costs to new clauses?

Systematic MaxSAT solvers are significantly less scalable than systematic SAT solvers

e.g., verification instance **`cmu-bmc-barrel6.cnf`**:

- best MaxSAT solvers need 20-30 minutes
- MiniSat needs ~2 seconds

36

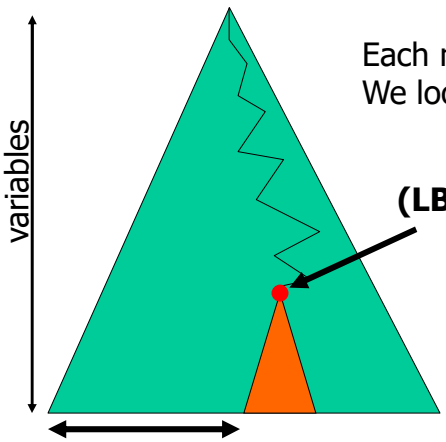
36

Branch&Bound for Hybrid MaxSAT

proc *HyMaxSATBB*(*H, F, σ, soln*)
// return the minimal cost of any interpretation
1 *res* := *BCP*(*H*);
2 **if** (*res* = *false*) **return** *false*; // hard causes violated
3 // Assume *res* = (*U, S*)
4 $\sigma := \sigma \cup U$;
5 *F* := *simplifySoft*(*F*, σ);
6 **if** *soln* ≤ *lowerBound*(*F*) **return** *soln*; // cut by bound
7 *A* := *pickLiteral*($S \cup F$); // branching on *A*.
8 **if** (*A* = *nil*) **return** *countFalseClauseWeight*(*F*, σ);
9 *soln* := *min*(*soln, HyMaxSATBB*($S \cup \{(A)\}$, *F*, σ, *soln*));
10 **return** *min*(*soln, HyMaxSATBB*($S \cup \{(A)\}$, *F*, σ, *soln*));

37

Branch&Bound: DPLL-like search



Each node is a MaxSAT subproblem
We look for the minimal cost solution

(LB) Lower Bound

under estimation of the best solution in the sub-tree

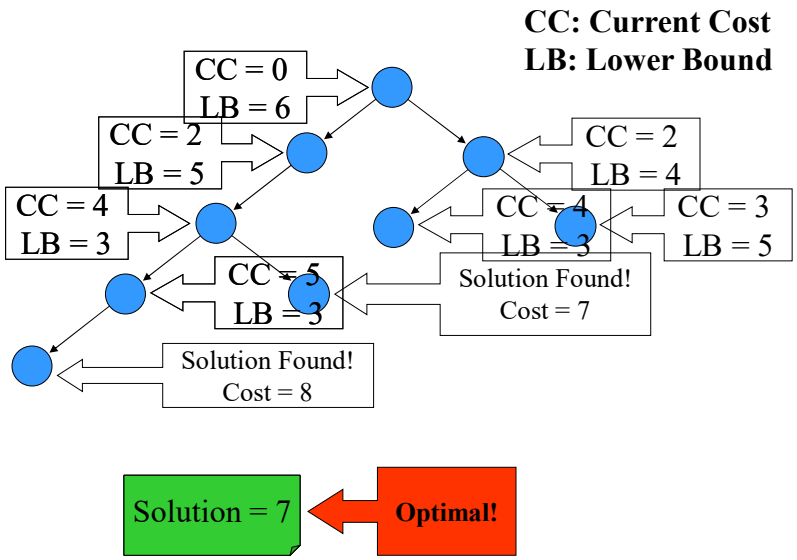
If LB ≥ CC then prune

(CC) Current Cost = best solution so far

38

38

Illustration: Branch&Bound Search



39

Simplification Rules for Soft Clauses

- **zero elimination**: Weighted clauses with zero weights can be discarded.
- **tautology elimination**: Valid clauses are always removed.
- **identical merge**: Identical clauses are merged into one.
  - $(\alpha; w_1), (\alpha; w_2) \rightarrow (\alpha; w_1 + w_2)$
- **unit clash**: Unit clauses with complement literals can be reduced to one
  - $(A; w_1), (A; w_2) \rightarrow (A; w_1 - w_2), (0; w_2)$
- **weighted resolution**

40

Weighted Resolution

$$(x \mid A;\ u),\ (\neg x \mid B;\ w) \ \Rightarrow \begin{cases} (A \mid B;\ m), \\ (x \mid A;\ u\text{-}m), \\ (\neg x \mid B;\ w\text{-}m), \\ (x \mid A \mid \neg B;\ m), \\ (\neg x \mid \neg A \mid B;\ m) \end{cases}$$

where $m = \min\{u, w\}$

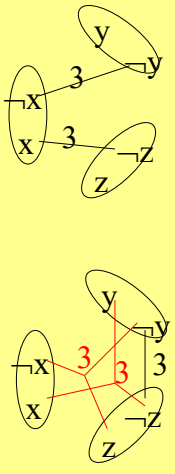
Example:
$(p \mid q;\ 2)$ and $(\neg p \mid q;\ 3)$ produces 5 clauses:
$(q \mid q;\ 2)$, $(p \mid q;\ 0)$, $(\neg p \mid q;\ 1)$, $(p \mid q \mid \neg q;\ 2)$, $(p \mid \neg q \mid q;\ 2)$.
which are simplified to $(q;\ 2)$, $(\neg p \mid q;\ 1)$.

41

41

Example: Meaning of weighted clauses

$$\left.\begin{array}{l} (x|y;\ 3), \\ (\neg x|z;\ 3) \end{array}\right\} = \begin{cases} (y|z;\ 3), \\ (x|y;\ 3\text{-}3), \\ (\neg x|z;\ 3\text{-}3), \\ (x|y|\neg z;\ 3), \\ (\neg x|\neg y|z;\ 3) \end{cases}$$



42

42

Linear Programming (LP) for Low Bound of Hybrid MaxSAT

	a	$\wedge$	(C1)
	$\neg a \vee b$	$\wedge$	(C2)
4:	$\neg b \vee c$	$\wedge$	(C3)
2:	$\neg c \vee d$	$\wedge$	(C4)
7:	$\neg d$		(C5)

=

Subject to C1
C2

Minimize $4\times\overline{C3}+2\times\overline{C4}+7\times\overline{C5}$

Linear Programming

Minimize $4\times C3+2\times C4+7\times C5$
Subject to $a \geq 1$,
$(1 - a) + b \geq 1$,
$1 - b) + c \geq$ **C3**,
$(1 - c) + d \geq$ **C4**,
$(1 - d) \geq$ **C5**,
$0 \leq a, b, c, d,$ **C3, C4, C5** ≤ 1

Integer Programming

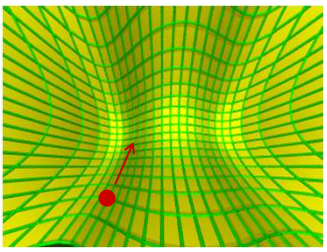
Minimize $4\times C3+2\times C4+7\times C5$
Subject to $a \geq 1$,
$(1 - a) + b \geq 1$,
$1 - b) + c \geq$ **C3**,
$(1 - c) + d \geq$ **C4**,
$(1 - d) \geq$ **C5**,
$a, b, c, d,$ **C3, C4, C5** $\in \{ 0, 1 \}$

43

43

Local Search: "Natural" for MaxSAT?

- Local search methods walk on a landscape composed of truth assignments
  - Height = number of unsatisfied constraints
- At every point in time, have a (sub-optimal) MaxSAT solution!
  - "anytime solution" to MaxSAT
- *Bottleneck*: local minima, just as in local search for SAT



Ongoing research topics

44

44