Decision Heuristics – Overview and Examples

Naïve Decision - Static

Heuristics

- Static variable order $x < y < z$, try positive assignments first.

Properties

- To detect UNSAT, all assignments need to be checked.
- For SAT, variable and sign ordering might strongly influence running time.

Example: Static Decision
Find a satisfying assignment for the formula: $(\neg x \vee y \vee z) \wedge (y \vee \neg z) \wedge (\neg x \vee \neg y)$ *Use the static variable order* $z < y < x$ *and try negative assignments first.*
This heuristic is simple. 1. Select $\alpha(z) = \alpha(y) = \alpha(x) = 0$, this directly satisfies the formula and we're done. (If we weren't done, we would continue with $\alpha(z) = 1, \alpha(y) = \alpha(x) = 0$)

Jeroslow-Wang Method – Static

Heuristics

1. For each literal l compute $J(l) = \sum_{\text{clause } c \text{ in the CNF containing } l} 2^{-|c|}$
2. Choose a literal l that maximizes $J(l)$

Properties

- Gives exponentially higher weights to literals in shorter clauses.

Example: Jeroslow-Wang Method
Find a satisfying assignment for the formula: $(\neg x \vee y \vee z) \wedge (y \vee \neg z) \wedge (\neg x \vee \neg y)$ *Use the Jeroslow-Wang Method and the fallback literal order* $\neg x < x < \neg z < z < \neg y < y$.
1. Compute $J(l)$ for each clause: $J(x) = 0$ $J(y) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$ $J(z) = \frac{1}{8}$ $J(\neg x) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$ $J(\neg y) = \frac{1}{4}$ $J(\neg z) = \frac{1}{4}$ 2. Since we have a tie, we use the fallback literal order and decide for $\neg x$. Afterwards, we decide for y, because it has the highest $J(\quad)$ value (since we've already assigned a value for $\neg c$) Finally, we assign $\neg z$ with same reasoning. Luckily, the assignment instantly satisfies the formula and we're done.

Dynamic Largest Individual Sum (DLIS) – Dynamic

Heuristics

1. Choose an assignment that increases the number of satisfied clauses the most.
2. For each literal l, let C_l be the number of unresolved clauses in which l appears and decide for the literal with highest C_l. In case of a tie, we use the fall-back variable ordering.

Properties

- Can be quite fast because we try to satisfy as many clauses with as few iterations as possible.

Example: Static Decision
Construct the decision tree for the formula: $(\neg x \vee y \vee z) \wedge (y \vee \neg z) \wedge (\neg x \vee \neg y)$ *using DLIS.* *The fallback literal order is defined by:* $\neg x < x < \neg z < z < \neg y < y$
1. Counters: $C_x = 0, C_{\neg x} = 2, C_y = 2, C_{\neg y} = 1, C_z = 1, C_{\neg z} = 1$ Because we have a tie, we decide for $\neg x$ (fallback literal order) $(\neg x \vee y \vee z) \wedge (y \vee \neg z) \wedge (\neg x \vee \neg y)$ 2. Counters: $C_x = 0, C_{\neg x} = 0, C_y = 1, C_{\neg y} = 0, C_z = 0, C_{\neg z} = 1$: Because we have a tie, we decide for $\neg z$ (fallback literal order) $(\neg x \vee y \vee z) \wedge (y \vee \neg z) \wedge (\neg x \vee \neg y)$ 3. Even though our formula is already SAT, we must assign a value to y. Our counters are all 0, so according to the fallback literal order, we decide for $\neg y$ (If we weren't done, we would continue with $\alpha(z) = 1, \alpha(y) = \alpha(x) = 0$)

Variable State Independent Decaying Sum (VSIDS) – Dynamic

Heuristics

1. Each variable (in each polarity) has a *counter* initialized to 0.
2. When resolution gets applied to a clause, the *counters of its literals are increased*.
3. Decision: The unassigned variable with the *highest counter* is chosen.
4. Periodically, all the counters are *divided* by a constant.

Properties

- Gives priority to variables involved in recent conflicts.
- Updates are needed only when adding new conflict clauses ⇒ Decision made in constant time

Example: DPLL + CDCL SAT Solving using VSIDS

Apply the DPLL + CDCL SAT Solving Algorithm using VSIDS as a decision heuristic and assign false to decision variables. Furthermore, we use watched literals to speed up propagation. The fallback variable ordering is $x_1 < x_2 < x_3 < x_4$:

$$\varphi = \underbrace{x_1 \vee x_2 \vee x_4}_{c_1} \wedge \underbrace{x_2 \vee \neg x_4}_{c_2} \wedge \underbrace{x_1 \vee \neg x_2 \vee x_4}_{c_3} \wedge \underbrace{x_3 \vee \neg x_4}_{c_4}$$

$DL0$: - (no trivial clauses)

Watch Lists	x_1	$\neg x_1$	x_2	$\neg x_2$	x_3	$\neg x_3$	x_4	$\neg x_4$
	c_1, c_3		c_1, c_2	c_3	c_4			c_2, c_4

Counter (increment = 1): $C(x_1) = 0,\ C(x_2) = 0,\ C(x_3) = 0,\ C(x_4) = 0$

$DL1$: $\neg x_1 : NULL$

- As all the counters are 0, we use the fallback ordering and decide for $x_1 = false$

- Propagate x_1 in $c_1: (x_1 \vee x_2 \vee x_4)$
  - Since $x_1 = false$, the watchlist must be updated: **Watch** $(\boldsymbol{x_2}, \boldsymbol{x_4})$ instead of (x_1, x_2)
- Propagate x_1 in $c_3: (x_2 \vee \neg x_4)$
  - Since $x_1 = false$, the watchlist must be updated: **Watch** $(\neg \boldsymbol{x_2}, \boldsymbol{x_4})$ instead of $(x_1, \neg x_2)$

Watch	x_1	$\neg x_1$	x_2	$\neg x_2$	x_3	$\neg x_3$	x_4	$\neg x_4$
Lists			c_1, c_2	c_3	c_4		c_1, c_3	c_2, c_4

$DL2: \neg x_2: NULL, \;\; x_4: c_1$

- As all counters are still 0, we use the fallback ordering and decide for $x_2 = false$
- Propagate $\neg x_2$ in $c_1: (x_1 \vee x_2 \vee x_4)$
  - Assign $x_4 = true$
- Propagate $\neg x_2$ in $c_2: (x_2 \vee \neg x_4)$
  - Assign $x_4 = false \Rightarrow$ Conflict! Apply conflict resolution.
    Resolve c_2 with c_1 and x_4
    $$\frac{c_2: (x_2 \vee \neg x_4) \quad c_1: (x_1 \vee x_2 \vee x_4)}{c_5: (x_1 \vee x_2)}$$
    Asserting clause, as only x_2 is from the current DL.
    $\Rightarrow$ Add asserting clause and *backtrack to $DL1$*

$$\varphi = \underbrace{x_1 \vee x_2 \vee x_4}_{c_1} \wedge \underbrace{x_2 \vee \neg x_4}_{c_2} \wedge \underbrace{x_1 \vee \neg x_2 \vee x_4}_{c_3} \wedge \underbrace{x_3 \vee \neg x_4}_{c_4} \wedge \underbrace{x_1 \vee x_2}_{c_5}$$

Watch	x_1	$\neg x_1$	x_2	$\neg x_2$	x_3	$\neg x_3$	x_4	$\neg x_4$
Lists	c_5		c_1, c_2, c_5	c_3	c_4		c_1, c_3	c_2, c_4

Counter (increment = 2): $C(x_1) = 1, \; C(x_2) = 1, \; C(x_3) = 0, \; C(x_4) = 1$

$DL1: \neg x_1: NULL, \;\; x_2: c_5, \;\; x_4: c_3, \;\; x_3: c_4$

- As all the counters are 0, we use the fallback ordering and decide for $x_1 = false$
- Propagate x_1 in $c_5: (x_1 \vee x_2)$
  - Assign $x_2 = true$
  - Propagate x_2 in $c_3: (x_1 \vee \neg x_2 \vee x_4)$
    - Assign $x_4 = true$
    - Propagate x_4 in $c_2: (x_2 \vee \neg x_4) \;\; \Rightarrow$ OK
    - Propagate x_4 in $c_4: (x_3 \vee \neg x_4)$
      - Assign $x_3 = true$
        All variables have been assigned without a conflict $\Rightarrow$ **SAT**

A satisfying assignment is given by $\alpha(x_1) = 0, \alpha(x_2) = \alpha(x_3) = \alpha(x_4) = 1$