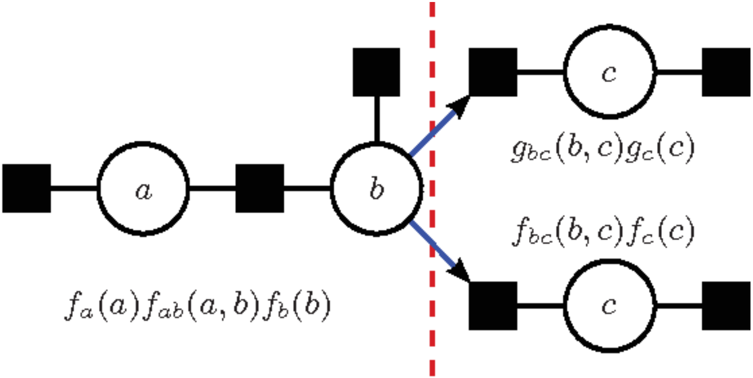


Belief Propagation

Table of Contents

SAT

- SAT formula in CNF

$$\mathcal{F} = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_3 \vee x_4)$$

  - Boolean variables $x_1, x_2, \ldots, x_n$
  - Negations $\overline{x_1}, \ldots, \overline{x_n}$
  - Clauses: Disjunction of variables and their negations
  - $\mathcal{F}$: Conjuction of clauses
- Is there an assignmnent of the variables that satisfies $\mathcal{F}$?
- How does the assignment look like?

Factor Graphs

Factor graphs represent a function's factorization

- Function $f(X)$ over variables $X = \{x_1, x_2, \ldots, x_n\}$
- Global function f factorizes to local functions

$$f(X) = \prod_{j=1}^{m} f_j(S_j)$$

- Local functions have smaller input $S_j \subset X$

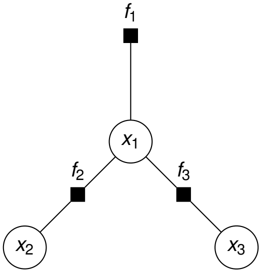
Factor Graphs

Factor graphs represent a function's factorization

- Two types of nodes
  - Variable nodes: represent variables
  - Factor nodes : represent local functions
- Edges connect variable and factor nodes
- Factor nodes are connected to all variable nodes of their input variables

Example

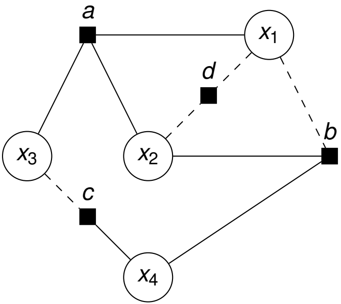
$$
\begin{aligned}
f(x_1, x_2, x_3) &= x_1^3 - x_1^2 x_2 + x_1^2 x_3 - x_1 x_2 x_3 \\
&= \underbrace{(x_1)}_{f_1(x_1)} * \underbrace{(x_1 - x_2)}_{f_2(x_1,x_2)} * \underbrace{(x_1 + x_3)}_{f_3(x_1,x_2)}
\end{aligned}
$$



Factor Graphs

Factor Graph of a CNF formula

$$\mathcal{F} = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2})$$

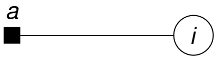


- $\mathcal{F}$ is a product of clauses
- Clauses $\cong$ local functions

Message Passing

Message Passing Algorithms on factor graphs

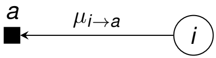
- Nodes communicate through messages
- Messages are passed over the graph's edges
- Two types of messages
  - $\mu_{i \to a}$ sent from factor a to variable i
  - $\mu_{a \to i}$ sent from variable i to factor a



Message Passing

Message Passing Algorithms on factor graphs

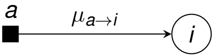
- Nodes communicate through messages
- Messages are passed over the graph's edges
- Two types of messages
  - $\mu_{i \to a}$ sent from factor a to variable i
  - $\mu_{a \to i}$ sent from variable i to factor a



Message Passing

Message Passing Algorithms on factor graphs

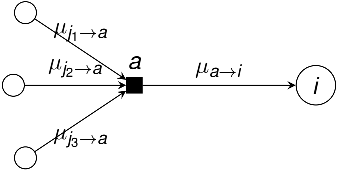
- Nodes communicate through messages
- Messages are passed over the graph's edges
- Two types of messages
  - $\mu_{i \to a}$ sent from factor a to variable i
  - $\mu_{a \to i}$ sent from variable i to factor a



Message Passing

Message Passing Algorithms on factor graphs

- Message $\mu_{a \to i}$ determined by incoming messages $\mu_{j \to a}$ from neighbours $j \neq i$
- Computation Rule depends on application

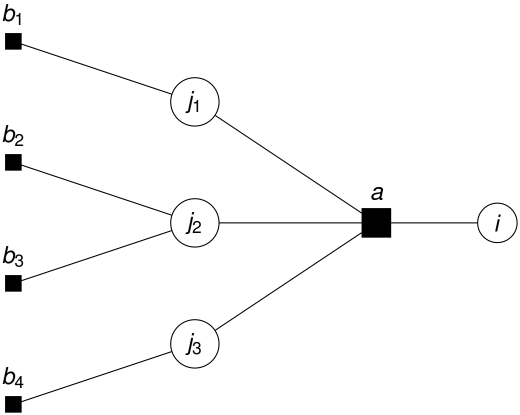


Message Passing

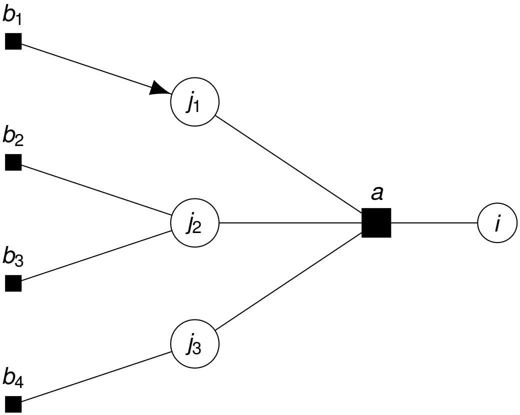
Message Passing Algorithms on trees

- Messages generated bottom up
- Leaves start sending messages
- Messages are propagated forward in the tree
- Does **not** work on graphs with cycles

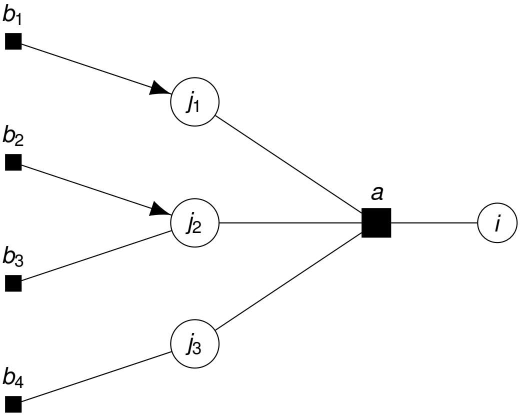
Example



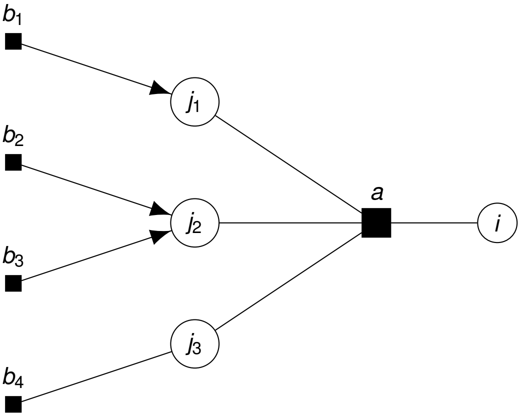
Example



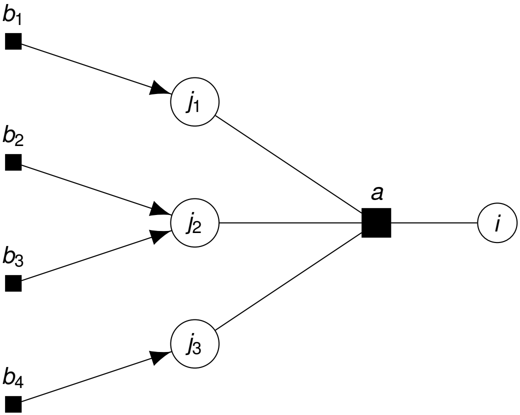
Example



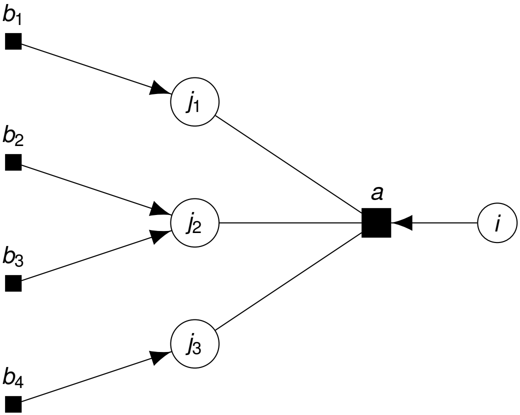
Example



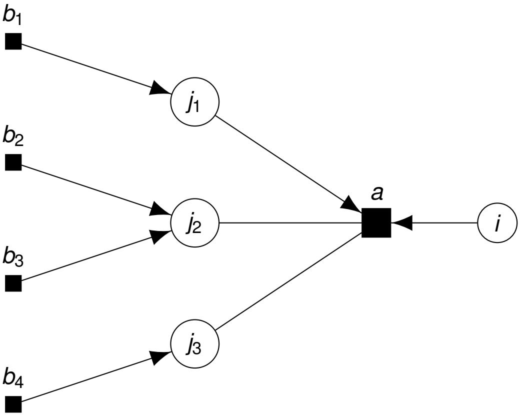
Example



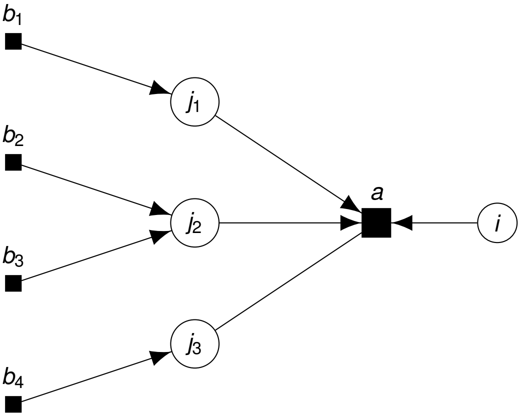
Example



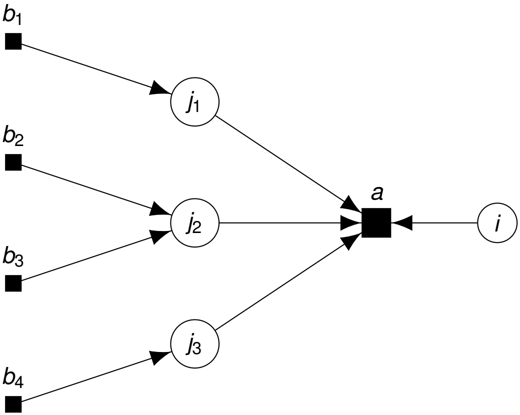
Example



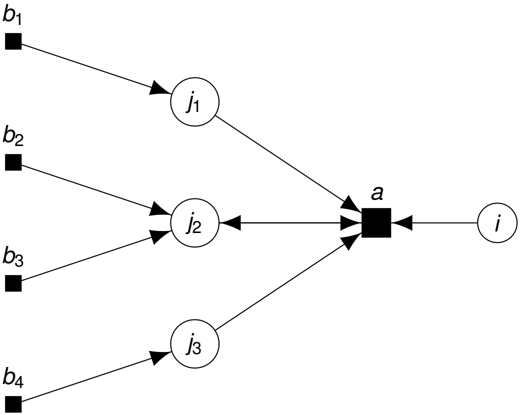
Example



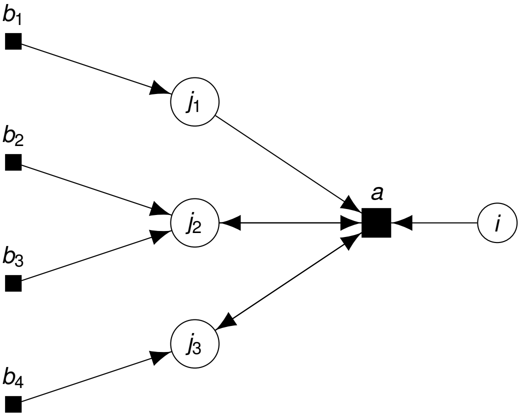
Example



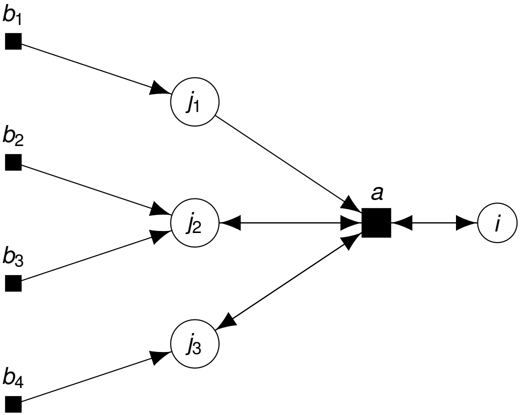
Example



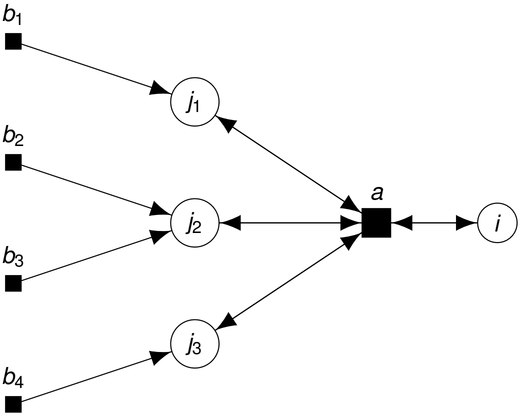
Example



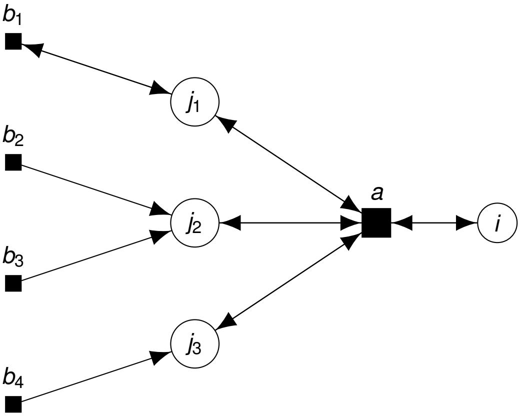
Example



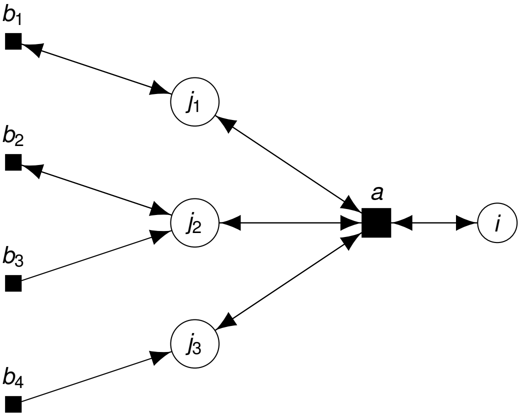
Example



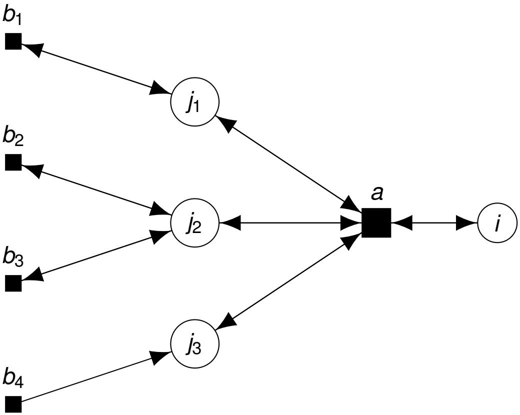
Example



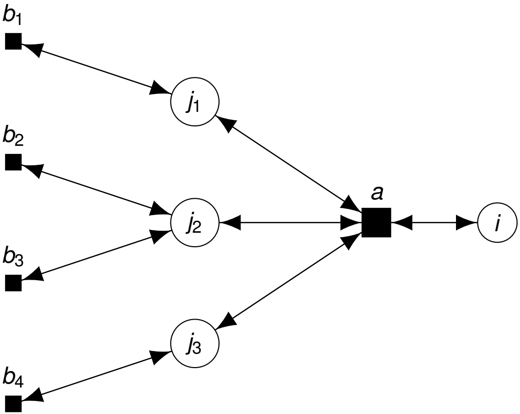
Example



Example



Example



Message Passing

- In general graphs: *Loopy* Message Passing
- Messages initialized with random values
- Each node repeatedly updates its outgoing messages based on the incoming ones
- Goal: *Convergence*
- Scheduling important

Message Passing

Generic Message Passing Algorithm

```
1. Randomly initialize all warnings μ_{i→a}, μ_{a→i}
2. For t = 0 to t_max
  2.1 Apply the update rule to all edges
      in random order
  2.2 If no message has changed goto 3
3. If t = t_max return UNGONVERGED
   Else     return the converged messages
```

Message Passing

- Loopy Message Passing converges on trees
- On cyclic graphs
  - no guarantee of convergence
  - no guarantee of correctness
- Can be used as heuristics
- In practice often correct

Warning Propagation Algorithm

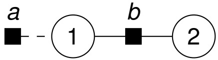
Apply Message Passing to SAT

- Input: Formula in CNF and its Factor Graph
- Output: Valid variable assignment

Warning Propagation Algorithm

Apply Message Passing to SAT

- Clause a can send a *warning* $u_{a \to i} \in \{0, 1\}$ to its variables i
- If $u^*_{a \to i} = 1$, i **must** satisfy a
  e.g $\mathcal{F} = (\overline{x_1}) \wedge (x_1 \vee x_2)$
  x_1 has to satisfy the first clause



- The clause a *fixes* the variable i to the value 0

Warning Propagation Algorithm

Apply Message Passing to SAT

- Variable i receives warnings $u_{b\to i}$
- i to sends to a its preferred value considering only the neighbours $b \neq a$
- *Cavity Field* $h_{i\to a}$
  - > 0, if i prefers the value 1
  - < 0 if i prefers the value 0
- Messages
  - Warnings $u_{a\to i} \in \{0, 1\}$
  - Cavity Fields $h_{i\to a} \in \mathbb{Z}$

Update Rule - Cavity Field

- Count how often i is fixed to 1 by clauses $b \neq a$

$$\sum_{b \in V_+(i) \setminus a} u_{b \to i}$$

- Count how often i is fixed to 0 by clauses $b \neq a$

$$\sum_{b \in V_-(i) \setminus a} u_{b \to i}$$

- Send the difference to a

$$h_{i \to a} = \sum_{b \in V_+(i) \setminus a} u_{b \to i} - \sum_{b \in V_-(i) \setminus a} u_{b \to i}$$

Update Rule - Warnings

- a sends a warning to i, if its variables $j \neq i$ prefer to violate a
- Define $J_j^a = \begin{cases} -1, & \text{if } x_j = 1 \text{ satisfies a} \\ 1, & \text{if } x_j = 0 \text{ satisfies a} \end{cases}$
- j prefers to violate a, if $J_j^a h_{j \to a} > 0$
- Define $\theta(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$

Update Rule - Warnings

- j prefers to violate a, if $J_j^a h_{j\to a} > 0$
- Define $\theta(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$
- Warnings can be computed as

$$u_{a\to i} = \prod_{j\in V(a)\setminus i} \theta(J_j^a h_{j\to a})$$

Update Rule - Summary

$$h_{i\to a} = \sum_{b\in V_+(i)\setminus a} u_{b\to i} - \sum_{b\in V_-(i)\setminus a} u_{b\to i}$$

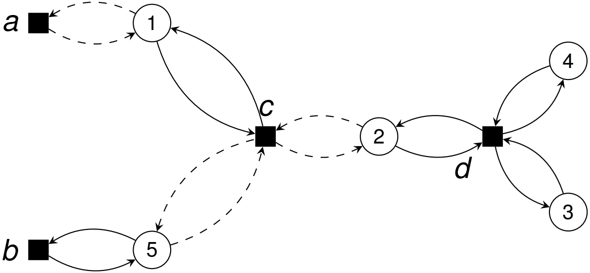
$$u_{a\to i} = \prod_{j\in V(a)\setminus i} \theta(J_j^a h_{j\to a})$$

Update Rule - Summary

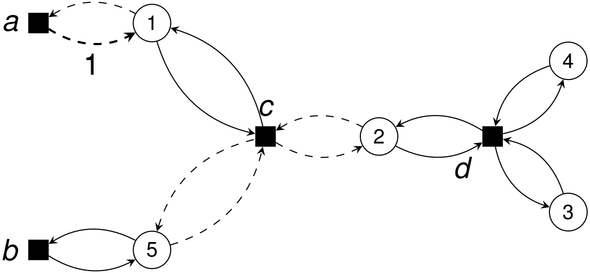
$$h_{i\to a} = -\sum_{b\in V_(i)\setminus a} J_i^b u_{b\to i}$$

$$u_{a\to i} = \prod_{j\in V(a)\setminus i} \theta(J_j^a h_{j\to a})$$

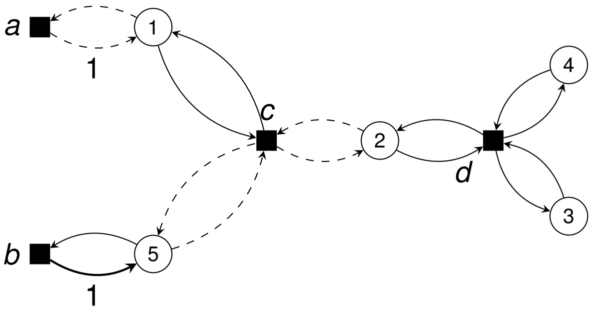
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



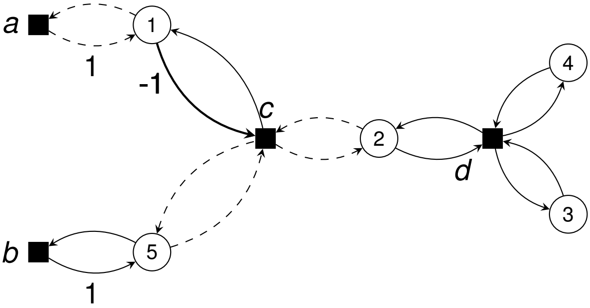
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



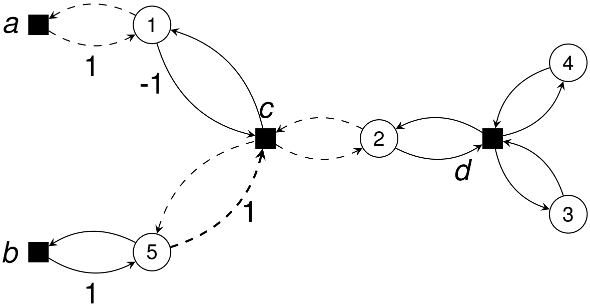
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



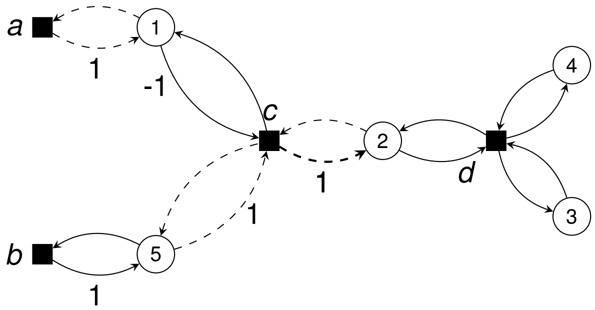
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



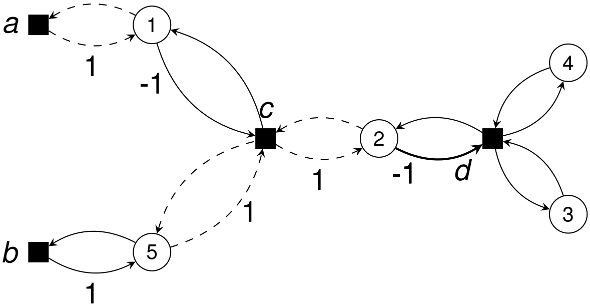
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



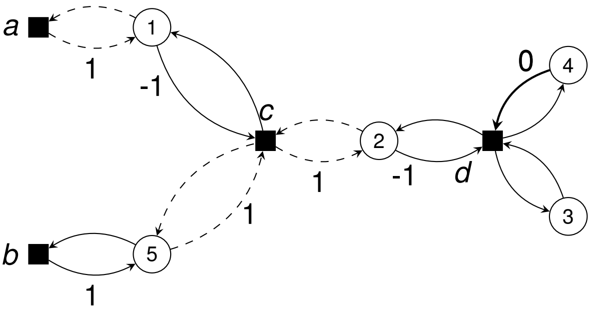
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



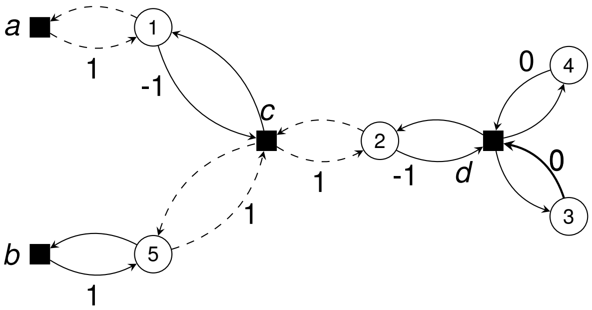
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



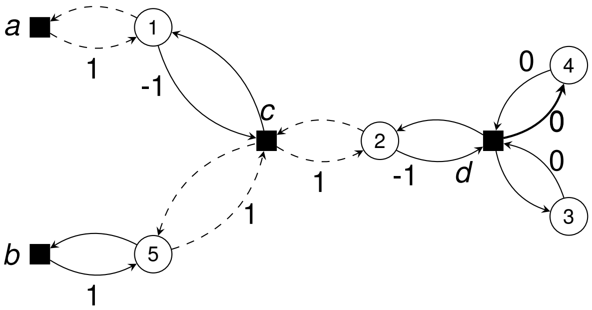
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



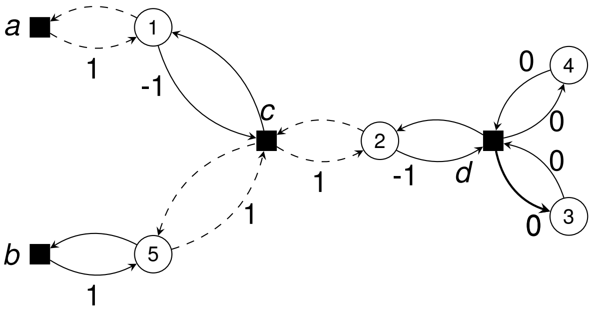
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



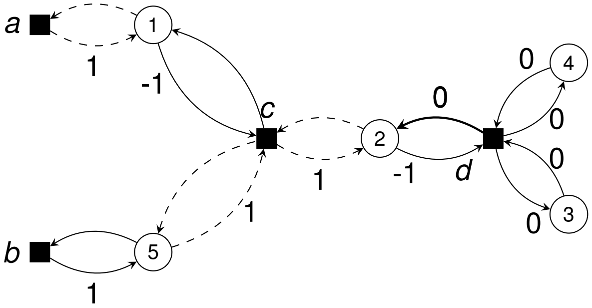
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



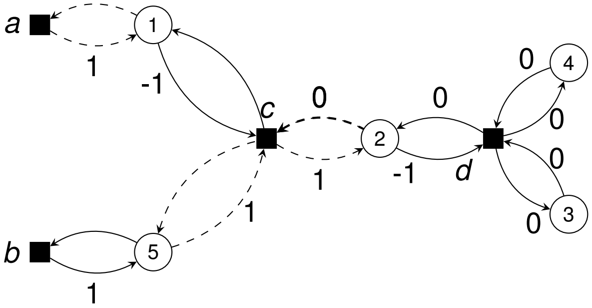
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



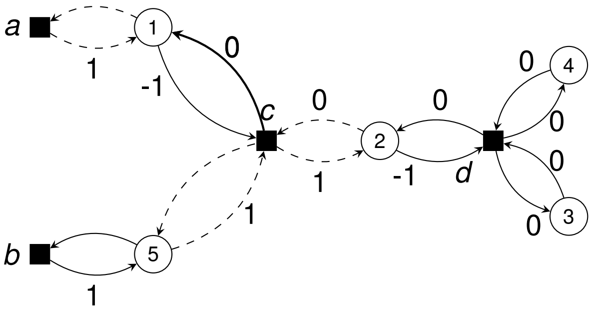
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



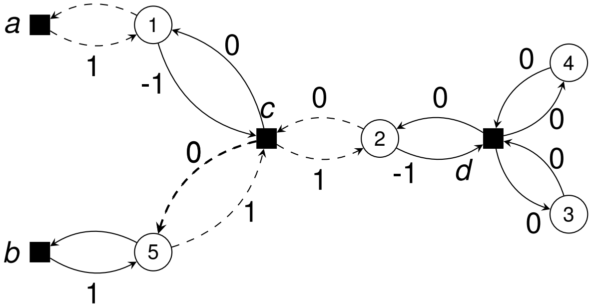
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



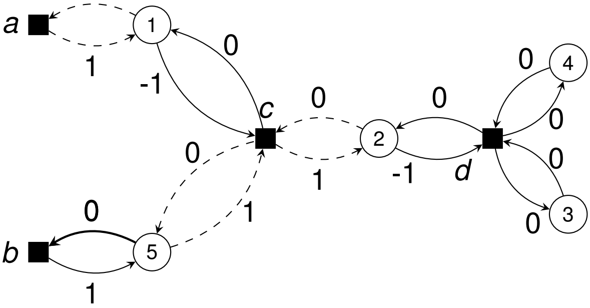
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



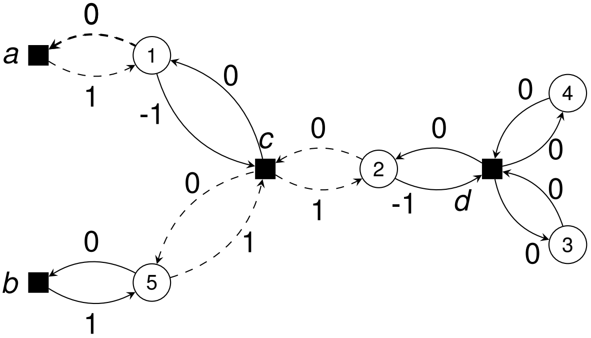
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



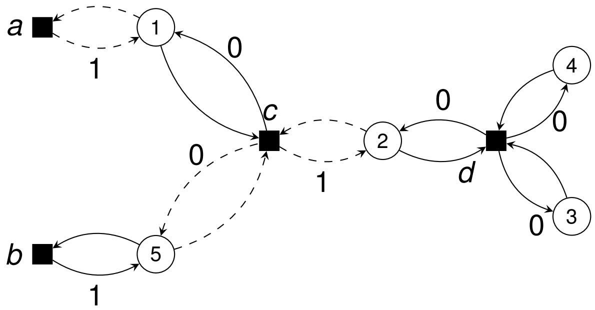
$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



$$\mathcal{F} = \underbrace{\overline{x_1}}_{a} \wedge \underbrace{x_5}_{b} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee \overline{x_5})}_{c} \wedge \underbrace{(x_2 \vee x_3 \vee x_4)}_{d}$$



Decimation Algorithm

- If a variable is fixed to different values, $\mathcal{F}$ is not satisfiable
- If not, the warnings lead to a partial assignment:
  - Assign each variable with active warnings the value it was fixed to
  - If all warnings are 0, assign a random value to a random variable

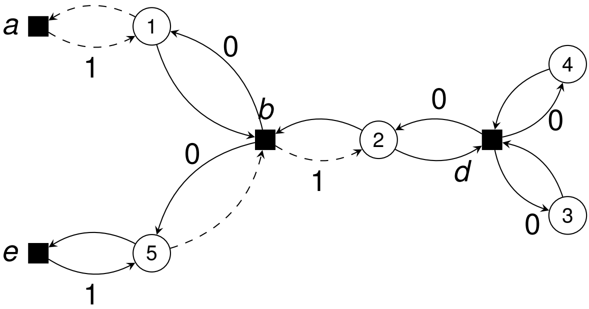
Message Passing

Warning Inspired Decimation Algorithm

```
1. While there are unassigned variables
2.      Run the WP Algorithm
3.      If WP does not converge
           return UNCONVERGED
4.      If at least one variable is fixed
         to different values return UNSAT
5.      Choose a partial assignment
6.      Clean the graph
7. Return the complete assignment
```

Example

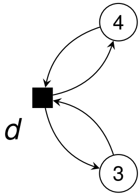
$$\mathcal{F} = \overline{x_1} \wedge x_5 \wedge (x_1 \vee \overline{x_2} \vee \overline{x_5}) \wedge (x_2 \vee x_3 \vee x_4)$$



Assign $x_1 = 0,\ x_2 = 0,\ x_5 = 1$

Example

- Cleaned formula: $\mathcal{F} = (x_3 \vee x_4)$
- Cleaned graph:



Belief Propagation Algorithm

- General Algorithm for approximating marginal probabilities on factor graphs
- Applied in many different areas
- Can be viewed as generalization of Warning Propagation

Belief Propagation Algorithm

Probability space built by all variable configurations $X = (x_1, \dots, x_n)$ of a formula $\mathcal{F}$.

- Uniform distribution over all assignments

$$P(X) = \frac{1}{2^n}$$

- Restriction on valid assignments

$$P(X \mid X \text{ satisfies } \mathcal{F}) = \frac{1}{\mathcal{N}} \prod_{a \in A} f_a(X)$$

- Goal: Compute $\mu_i := P(x_i = 1 \mid X \text{ satisfies } \mathcal{F})$

Messages

- WP: $u_{a \to i}$ is 1 if all neighbours $j \in V(a) \setminus i$ violate a
- BP: $\delta_{a \to i}$ is the *probability* for this event

Messages

- WP: $h_{i \to a}$: preferred value of *i*
- BP: $\gamma_{i \to a}$: *probability* for *i* to violate *a*

Decimation

Decimation Algorithm is based on computed marginals

- Similiar to Warning Propagation
- Run BP Algorithm
- Assign $x_i = 1$ if $\mu_i > 0.5$ and $x_i = 0$ otherwise
- Clean the graph

Conclusion