Survey Propagation for Random K-sat problems

Alex Olshevsky

6.454 Presentation

Outline

- Introduction to K-sat. Outline of the next three weeks.

- Algorithms for solving K-sat:
  - warning propagation
  - survey propagation
  - belief propagation

- Main result: survey propagation is a subcase of belief propagation

K-sat

- A clause with d variables is a the logical OR of the variables $x_1, \ldots, x_d$, or their negations. If $d = 3$, the following are clauses:

$$x_1 \vee x_2 \vee x_3$$

$$x_1 \vee \bar{x}_2 \vee x_3$$

$$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$$

where "$\vee$" is the logical OR symbol and $\bar{x}$ is the negation of x.

K-sat

- K-sat: determine whether

$$P = C_1 \wedge C_2 \wedge \cdots \wedge C_k,$$

(where "$\wedge$" is the logical AND symbol and each C_i is a clause with K variables) is satisfiable for some assignment of the input variables x_i.

- Example:

$$(x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee \bar{x}_3)$$

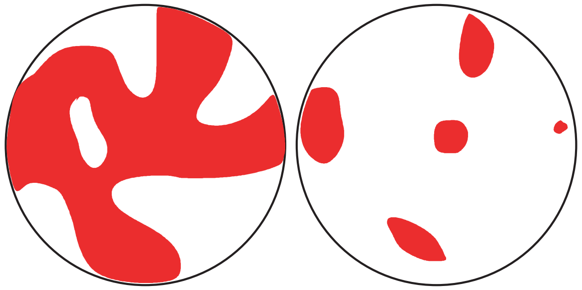
is satisfiable. We can set $x_1 = 1$, $x_2 = 0$, $x_3 = 0$. On the other hand

$$(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

is not satisfiable.

Experimental Results for Random K-sat

- A random K-sat problem is generated by choosing each C_i, $i = 1, \ldots, n$ via uniform sampling from the space of all $\binom{n}{K}$ subsets size of K of the set $\{x_1, \ldots, x_n\}$, and negating each variable with probability $1/2$.



-

Let $\alpha = k/n$ be fixed, and let n be large. Then, for $K = 3$:

1. There exists a constant α_c, such that for $\alpha < \alpha_c$, the problem is almost always satisfiable and for $\alpha > \alpha_c$, the problem is almost always unsatisfiable.
2. There exists a constant α_d, smaller than α_c, such that for $\alpha \in (0, \alpha_d)$, the set of solutions almost always forms a connected a cluster.
   On the other hand, for $\alpha \in (\alpha_d, \alpha_c)$, the set of solutions almost always consists of disconnected clusters.

Why study K-sat problems?

From a CS perspective:

- They have some applications in AI.
- More importantly, K-sat is an NP-complete problem. Any algorithm which solves K-sat will solve many other problems (like integer programming) with at most a polynomial-time slowdown.
- Average-case complexity of K-sat relates to the hardness of approximation of many combinatorial problems in NP.

Average Case Complexity of K-sat

- A refutation algorithm for a K-sat problem: given an instance of the problem, announces whether it is satisfiable or not.

- **Never** wrong if the problem is satisfiable.

- On a random K-sat problem, correct most of the time.

- **Hypothesis:** Even when k/n is an arbitrarily large constant independent of n, there is no polynomial time refutation algorithm for 3-sat.

  – Uriel Feige, "Relations between average case complexity and approximation complexity," STOC 2002.

- The hypothesis implies inapproximability bounds for a large number of combinatorial problems (for example min bisection).

Spin Glasses

- K-sat can be seen as a simple case of "spin glass" models for disordered systems.
- Spin glasses: disordered magnetic systems.

Spin Glasses

- $\Sigma_N = \{0,1\}^N$ of n-letter words from the alphabet $\{0,1\}$. Each $\sigma \in \Sigma_N$ is a representation of N spins which are either $+1$ or -1.

- Let $H_N(\sigma)$ be a real valued function representing the potential of the configuration σ.

- Gibbs probability measure on Σ_N defined by

$$p(\sigma) = \frac{1}{Z} e^{-H_N(\sigma)/T}$$

  where Z is a normalizing constant and T is a parameter called temperature.

- As $T \to 0$, we have that $p(\sigma)$ approaches a uniform measure over all configurations minimizing H_N.

Spin Glass Models

$$H = \sum_{i,k} J_{ik} \sigma_i \sigma_k$$

- Sherrington-Kirkpatrick model: All J's are independently Gaussian, with mean 0 and variance $\sqrt{N}$.

- Edwards-Anderson model: each σ_i has a location in R^D, and J_{ik} is nonzero only if i, k are neighbors. In this case, J_{ik} is Gaussian with mean 0 and variance $\sqrt{D}$.

- Long-range Edwards-Anderson model: J_{ik} is nonzero if and only if the distance between i and k is at most R. If i, k have distance $r \leq R$, then J_{ik} has mean 0 and variance $1/R^{D/2}$.

Spin Glass Models and K-sat

- Consider Σ_N to be the set of assignments to a K-sat problem, and let $H_N(\sigma)$ be the number of violated constraints in assignment σ.

- $H_N(\sigma)$ is minimized at the solution(s) of the K-sat problem

- As $T \to 0$,
$$p(\sigma) = \frac{1}{Z} e^{-H_N(\sigma)/T}$$
is a uniform measure over all the solutions of the K-sat problem.

- K-sat can be cast as a zero-temperature limit of a simple defined spin glass system.

Outline of the Presentations

"Two main categories of questions ... One type is algorithmic (e.g., finding an algorithm that decides whether an instance is SAT or UNSAT, or that tries to minimize the number of violated constraints). Another one is more theoretical and deals with random instances for which one wants to predict the typical behavior (e.g., phase transitions and structure of the solution space)."

-M. Mezard, G. Parisi, R. Zecchina.

1. Today: survey propagation, a way to solve Ksat problems below the threshold.

2. Next week: phase transition between satisfiability and unsatisfiability.

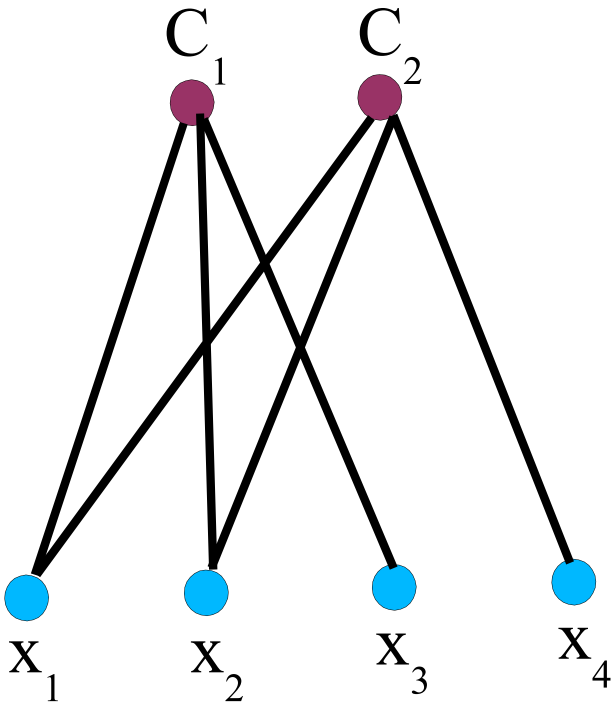
3. Two weeks from today: Nonexistence of solution clusters at low density and existence of solution clusters at moderate density below the threshold.

Warning propagation: an algorithm for random K-sat problems

- Notation: Letters a,b,c,... will be used for constraints, letters i,j,k,... will be used for variables.

- Factor graph: a node for every variable and every constraint, and a link (i, a) if constraint a involves variable i.

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4)$$



Warning propagation: an algorithm for random K-sat problems II

- Warning propagation is a message passing algorithm on the factor graph. At each stage, messages get passed from constraints to nodes, and nodes to constraints.

- Messages from variables to constraints are either "s," "u," or "*"

- Messages from constraints to variables are either "w" or empty.

Message update rules

- A constraint sends a "w" to a variable if
  - That variable is the **only** variable in the constraint
  - All other variables in the constraint have sent a "u" to the constraint

  Else, it sends an empty message.

- Some notation:
  - $C_a^s(i)$ is the set of constraints whose preferred assignment for i is the same as the preferred assignment of a (not including a itself).
  - $C_a^u(i)$ is the set of constraints whose preferred assignment for i is the **opposite** the preferred assignment of a.

- A node i sends an "u" to constraint a if it has received more warnings from nodes in $C_a^u(i)$ than in $C_a^s(i)$. In the reverse case, it sends and "s."

  If a node received no warnings, it sends a "*" message.

Performance of Warning Propagation

- Experimental observation: warning propagation seems to work on random K-sat problems of low density, but fails as density gets larger - even though the problem remains solvable.

- Warning propagation works exactly if the factor graph is a tree. Define

$$\begin{aligned} c_i &= 1 \text{ if at the fixed point variable } i \text{ receives warnings from two constraints with} \\ &\quad \text{different preferred values for } i \\ &= 0 \text{ if at the fixed point variable } i \text{ receives no warnings, or all the warnings} \\ &\quad \text{it receives are from constraints with the same preferred value for } i \end{aligned}$$

Theorem. Consider an instance of the K-sat problem on N variables for which the factor graph is a tree. Then at time N the warning propagation algorithm converges to a fixed point. Moreover, if any c_i are nonzero, the problem is unsatisfiable; if all the c_i are zero, the problem is satisfiable.

Proof

- Let $T_{a \to i}$ be the subgraph defined by removing the edge (a, i) and picking the component containing a.

- The level of the edge (a, i) is defined as the **largest** distance from a to a leaf in $T_{a \to i}$.

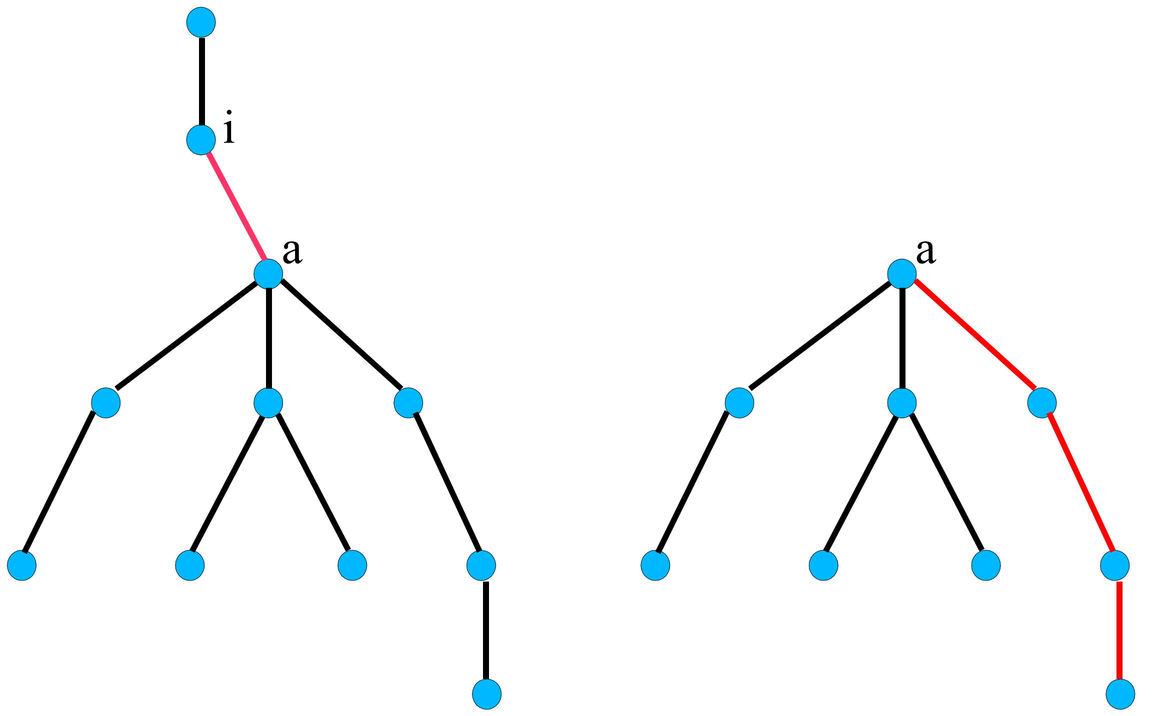
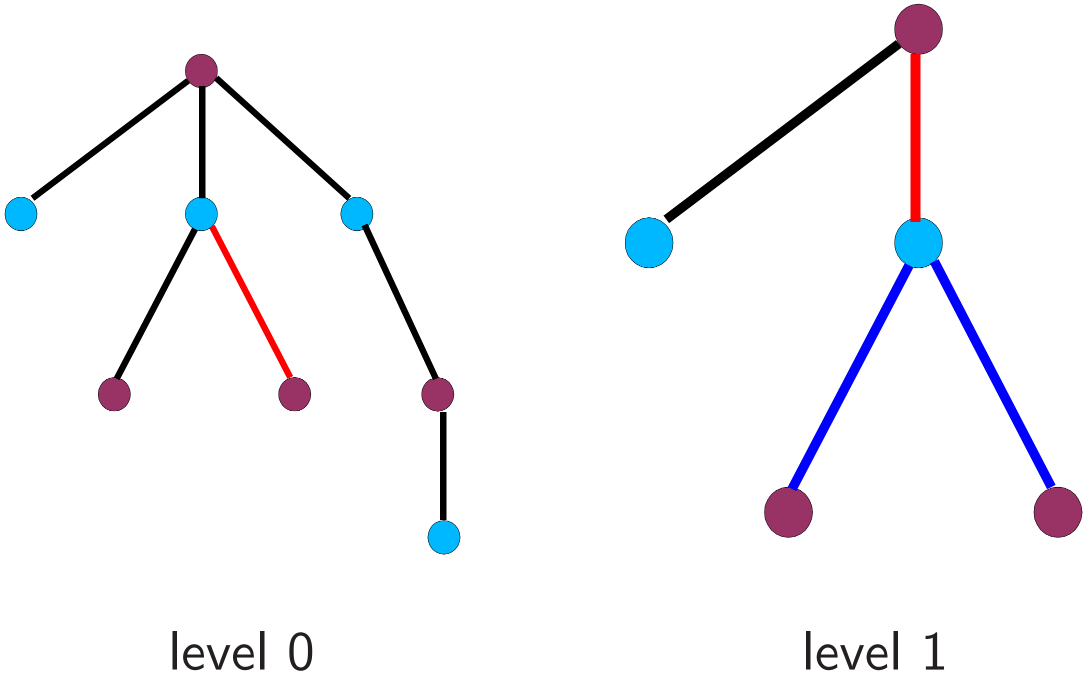


Figure 1: The level of the edge (a, i) is 3.

Fixed point

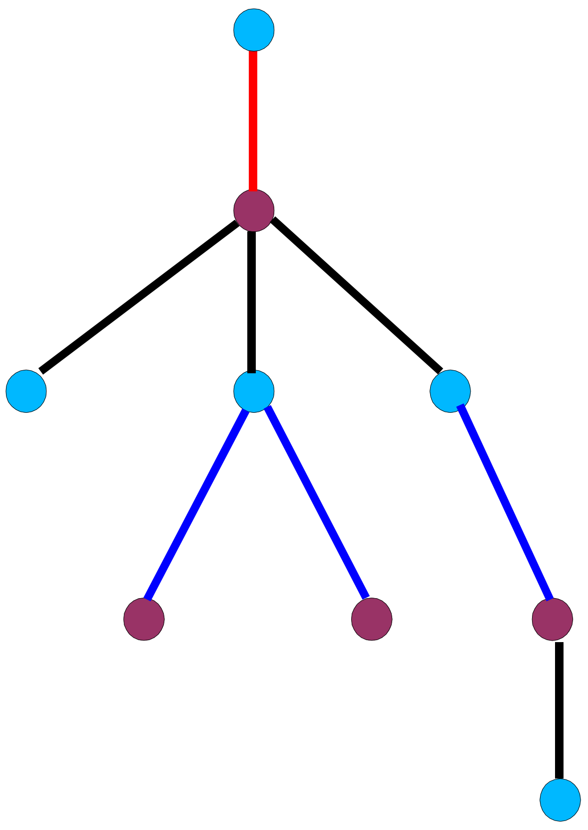
- First, proof of the fixed point.
- On an edge with level 0, messages are constant after time 1.
- On an edge with level 1, messages are constant after time 2.



- Inductive hypothesis: at level r, messages are constant after time $1 + r$.

Fixed point

- Claim: messages at level r are completely by determined by messages with level at most $r-2$.



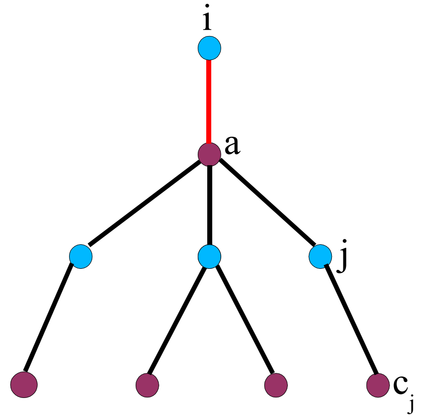
- Blue edges have level at most $r-2$. This is because all other constraints "below" the root are on a path from the root to some leaf.

Warnings

Lemma: If $u^*_{a \to i} = w$, then clause a is violated in the K-sat problem defined by the subgraph $T_{a \to i}$.

- Induction. True for when level is zero.

- Suppose $u^*_{a \to i} = w$. Consider a neighbor variable node j. At the fixed point, j must have received a warning from some constraint c_j, leading it to send "u" to constraint a. By the inductive hypothesis, the K-sat problem corresponding to the subgraph $T_{c_j \to j}$ is unsatisfiable! So, value of node j is fixed by constraint c_j to satisfy the problem.



Contradiction numbers

$$
\begin{aligned}
c_i &= 1 \text{ if at the fixed point variable } i \text{ receives warnings from two constraints with} \\
&\quad \text{different preferred values for } i \\
&= 0 \text{ if at the fixed point variable } i \text{ receives no warnings, or all the warnings} \\
&\quad \text{it receives are from constraints with the same preferred value for } i
\end{aligned}
$$

- After we showed that a warning from a to i fixes the value of i, it follows that a nonzero contradiction number implies the problem is satisfiable.
- What about the reverse? If the contradiction numbers are all zero at the fixed point of warning propagation, is the problem satisfiable?
- The answer is yes. Next: how to produce a satisfying assignment from the fixed point of warning propagation.

Contradiction numbers

Algorithm:

- Fix the value of any node that has received the warning. Clean the graph, i.e. remove those nodes and all the constraints that are now satisfied. This breaks up the graph into disjoint trees.
- Only nodes with "*" values will be left.
- Pick a node at random, and fix it to a value. Clean the graph again.
- Initialize warning propagation on the graph at its previous fixed point.
- Nodes adjacent to a fixed variable may send warnings! But every variable node may receive at most one warning, **from the neighbor on the path to a constraint adjacent to a fixed variable**.

Survey Propagation: Statistics of Warning Propagation With Some Unrealistic Assumptions

- Denote the probabilities of sending a "u," "s," and "*" on link (i,a) by $\Pi^u_{i\to a}$,$\Pi^s_{i\to a}$, and $\Pi^*_{i\to a}$; and denote the probability of warning on link (i,a) as $\eta_{a\to i}$.

- **Assuming all warning events are independent,** we have that a warning is sent on a link if and only if the constraint receives "u"s from all other links.

$$\eta_{a\to i} = \prod_{j\in N(a)-\{i\}} \frac{\Pi^u_{j\to a}}{\Pi^u_{j\to a} + \Pi^s_{j\to a} + \Pi^*_{j\to a}}$$

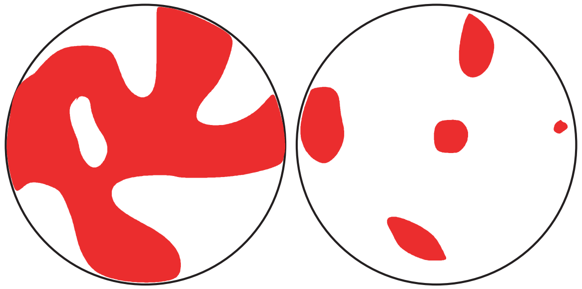
  where $N(a)$ is the set of neighbors of a in the factor graph.

- A link will send a $*$ message if and only if it will get no warnings at all:

$$\Pi^*_{i\to a} = \prod_{b\in N(i)-\{a\}} (1-\eta_{b\to i})$$

Survey Propagation: Statistics of Warning Propagation With Some Unrealistic Assumptions

- Another assumption: no node gets contradictory warnings. Intuitively, we assume that warning propagation stays within a cluster.



- Node i sends a "u" to constrain a only if it receives warnings from constraints that disagree with a.

$$\Pi^u_{i\to a} = [\prod_{b\in C^s_a(i)} (1-\eta_{a\to i})][1 - \prod_{b\in C^u_a(i)} (1-\eta_{a\to i})]$$

$$\Pi^s_{i\to a} = [\prod_{b\in C^u_a(i)} (1-\eta_{a\to i})][1 - \prod_{b\in C^s_a(i)} (1-\eta_{a\to i})]$$

A Decimation Based Approach

- Use survey propagation to fix one variable at a time.
- Experimental results: correct answer for moderately dense random K-sat problems, perhaps up to the threshold?
- Big question: Why does it work?
- Next: survey propagation as belief propagation.

Belief propagation

- $g(z_1, \ldots, z_n)$ is a function of the variables $z_1, \ldots, z_n$, which take values in some discrete set.

- Suppose $g(z_1, \ldots, z_n)$ factors into a product of "local functions" f_j, where each f_j is defined on some subset of $\{z_1, \ldots, z_n\}$:

$$g(z_1, \ldots, z_n) = \prod_j f_j(Z_j)$$

  Z_j is a subset of $\{z_1, \ldots, z_n\}$, and $f_j(Z_j)$ is a function of the variables in the set Z_j.

- We will use the "not-sum" $\sum_{\sim\{z_i\}}$ to indicate that the sum is over every variable except z_i, i.e.

$$\sum_{\sim\{z_i\}} g(z_1, \ldots, z_n) = \sum_{z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n} g(z_1, \ldots, z_n)$$

Belief propagation

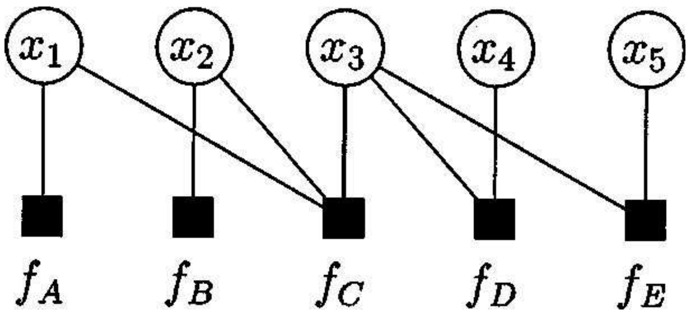
- Our goal in this section is to efficiently compute the "marginal functions"

$$g_i(z_i) = \sum_{\sim\{z_i\}} g(z_1, \ldots, z_n)$$

- Factor graph: a node for every variable z_i, a node for every function f_j, and links between functions and the variables used by them.

- Example:

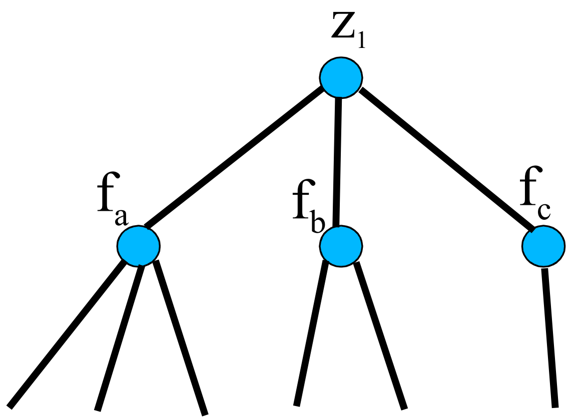
$$g(x_1, x_2, \ldots, x_5) = f_{\mathrm{A}}(x_1) f_{\mathrm{B}}(x_2) f_{\mathrm{C}}(x_1, x_2, x_3) f_{\mathrm{D}}(x_3, x_4) f_{\mathrm{E}}(x_3, x_5)$$



Belief Propagation

- Cartesian product lemma: for any two disjoint sets X,Y and for any two functions f, h

$$\sum_{x \in X, y \in Y} f(x)h(y) = (\sum_{x \in X} f(x))(\sum_{y \in Y} h(y))$$



- Let Z_a, Z_b, Z_c represent the variable sets used by the functions f_a, f_b, f_c. Let f_{ai}, $i = 1, 2, \ldots$ be all the functions in the branch of the tree under f_a, and let their variable sets be Z_{ai}; similarly for f_{bi}, f_{ci} and Z_{bi}, Z_{ci}.

- $$\sum_{\sim z_1} g(z_1, z_2, \ldots) = \sum_{\sim z_1} f_a(Z_a)(\prod f_{ai}(Z_{ai}))f_b(Z_b)(\prod f_{bi}(Z_{bi}))f_c(Z_c)(\prod f_{ci}(Z_{ci}))$$

Belief Propagation

$$\sum_{\sim z_1} g(z_1, z_2, \ldots) = [\sum_{\sim z_1} f_a(Z_a) \prod f_{ai}(Z_{ai})] \cdot [\sum_{\sim z_1} f_b(Z_b) \prod f_{bi}(Z_{bi})] \cdot [\sum_{\sim z_1} f_c(Z_c) \prod f_{ci}(Z_{ci})] \quad (1)$$

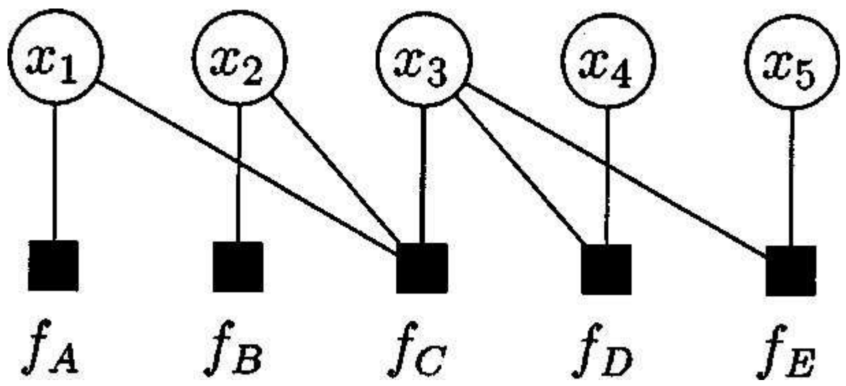
This factorization gives a "bottom-up" algorithm for computing $g_1(z_1)$, which computes the quantities $\sum_{\sim z_1} f_k(Z_k) \prod f_{ki}(Z_{ki})$ recursively for subbranches of the tree.

1. Each leaf variable node sends an identity message to its parent; each leaf function node sends a description of itself to its parent

2. When a variable node receives messages from all its children, it forms the product of them and sends them to its parent (where product is the simple pointwise product of functions).

3. When a function node f_i with parent z_i receives messages from all its children, it forms the product of these messages with $f_i(Z_i)$ and operates on the result with the $\sum_{\sim z_i}$ operator.

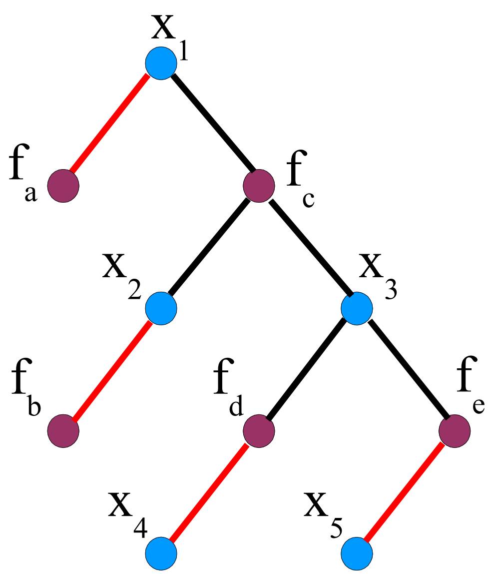
An Example



$$g(x_1, x_2, \ldots, x_5) = f_{\mathrm{A}}(x_1) f_{\mathrm{B}}(x_2) f_{\mathrm{C}}(x_1, x_2, x_3) f_{\mathrm{D}}(x_3, x_4) f_{\mathrm{E}}(x_3, x_5)$$

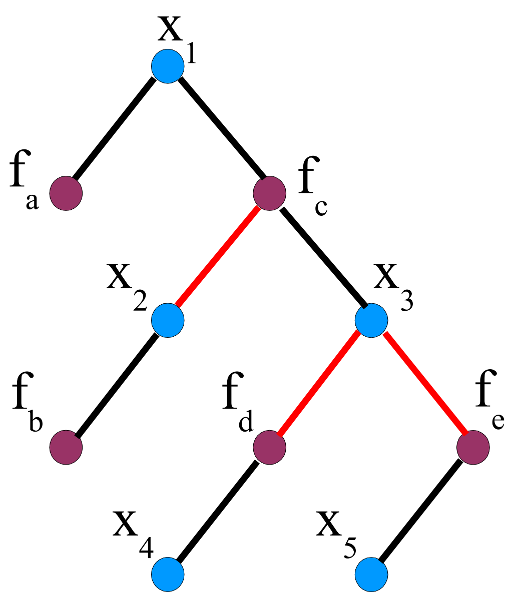
$$\begin{aligned} g_1(x_1) &= \sum_{x_2, x_3, x_4, x_5} g(x_1, x_2, x_3, x_4, x_5) \\ &= f_{\mathrm{A}}(x_1) \sum_{x_2} (f_{\mathrm{B}}(x_2) (\sum_{x_3} f_{\mathrm{C}}(x_1, x_2, x_3) (\sum_{x_4} f_{\mathrm{D}}(x_3, x_4)) (\sum_{x_5} f_{\mathrm{E}}(x_3, x_5)))) \end{aligned} \tag{2}$$

An Example



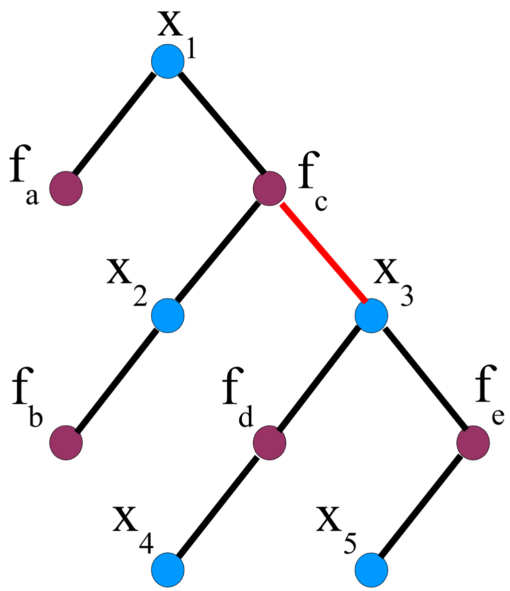
1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b,\ \mu_{a\to 1} = f_a.$

An Example



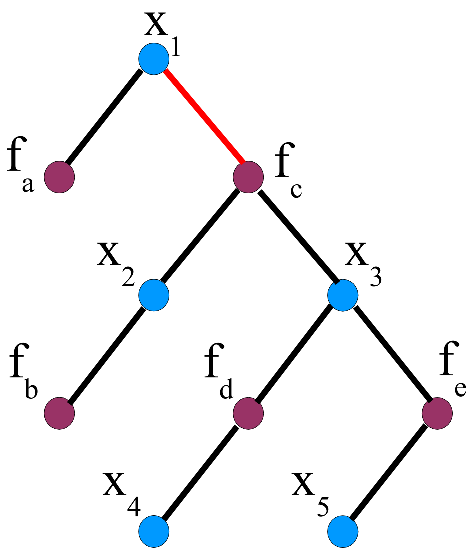
1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b,\ \mu_{a\to 1} = f_a.$
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5),\ \mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4),\ \mu_{2\to c} = f_b.$

An Example



1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b$, $\mu_{a\to 1} = f_a$.
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5)$, $\mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4)$, $\mu_{2\to c} = f_b$.
3. $\mu_{3\to c} = (\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5))$.

An Example



1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b, \ \mu_{a\to 1} = f_a.$
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5), \ \mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4), \ \mu_{2\to c} = f_b.$
3. $\mu_{3\to c} = (\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5)).$
4.

$$\begin{aligned}\mu_{c\to 1} &= \sum_{\sim x_1} f_c(x_1, x_2, x_3) f_b(x_2)(\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5)) \\ &= \sum_{x_2}(f_{\mathrm{b}}(x_2)(\sum_{x_3} f_{\mathrm{c}}(x_1, x_2, x_3)(\sum_{x_4} f_{\mathrm{d}}(x_3, x_4))(\sum_{x_5} f_{\mathrm{e}}(x_3, x_5))))\end{aligned}$$

An Example

1. Product of messages sent to x_1 is

$$f_{\mathrm{a}}(x_1)\sum_{x_2}(f_{\mathrm{b}}(x_2)(\sum_{x_3} f_{\mathrm{c}}(x_1,x_2,x_3)(\sum_{x_4} f_{\mathrm{d}}(x_3,x_4))(\sum_{x_5} f_{\mathrm{e}}(x_3,x_5))))$$

   which is $g_1(x_1)$.

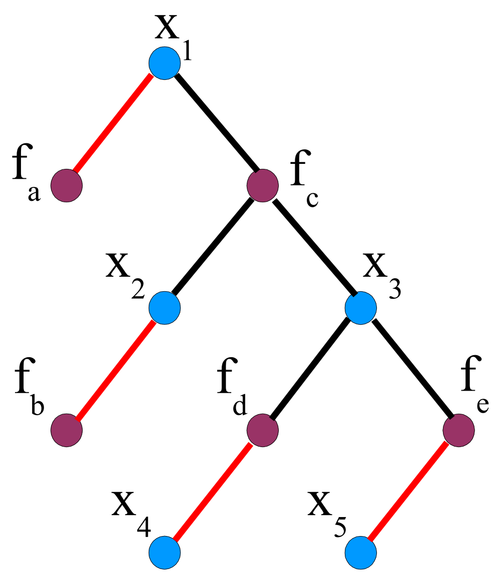
2. What if we want to compute all $g_i(x_i)$? We can implement a parallelized schedule of iterations:

$$\mu_{x_i\to f_j} = \prod_{h\in n(x_i)-\{f_j\}} \mu_{h\to x_i}$$

   and

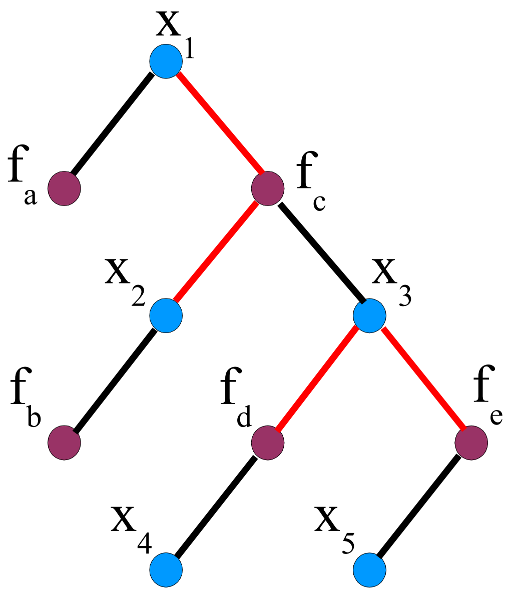
$$\mu_{f_j\to x_i} = \sum_{\sim x_i}(f_j(X_j) \prod_{y\in n(f_j)-\{x_i\}} \mu_{y\to f_j})$$

Parallelization: An Example



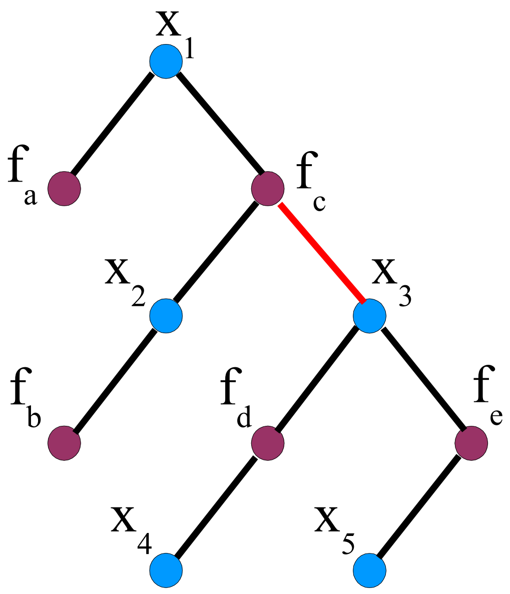
1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b$, $\mu_{a\to 1} = f_a$.

Parallelization: An Example



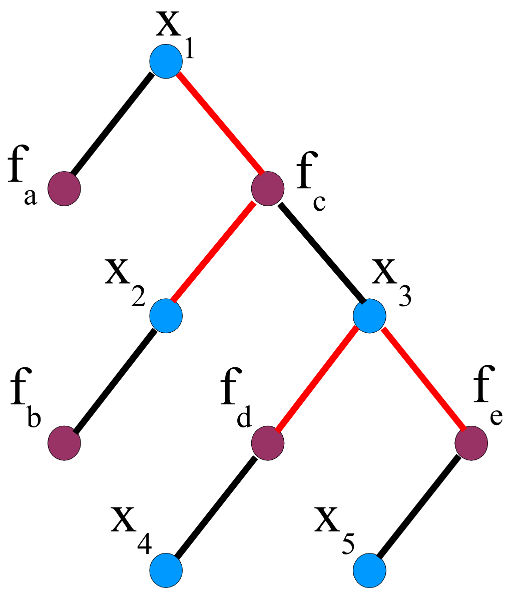
1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b$, $\mu_{a\to 1} = f_a$.
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5)$, $\mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4)$, $\mu_{2\to c} = f_b$, $\mu_{1\to c} = f_a$.

Parallelization: An Example



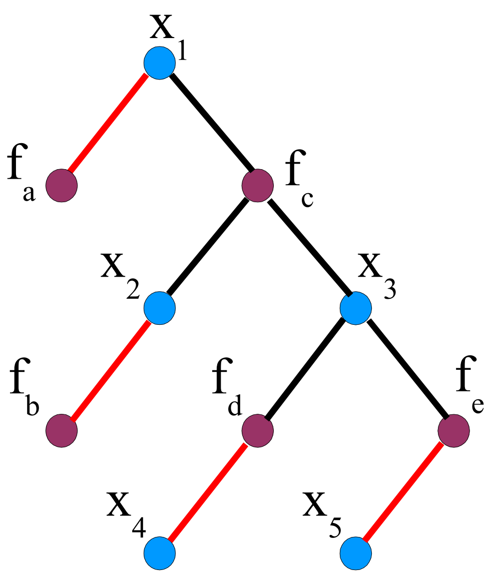
1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b$, $\mu_{a\to 1} = f_a$.
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5)$, $\mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4)$, $\mu_{2\to c} = f_b$, $\mu_{1\to c} = f_a$.
3. $\mu_{3\to c} = (\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5))$, $\mu_{c\to 3} = \sum_{x_1,x_2} f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3)$.

Parallelization: An Example



1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b, \mu_{a\to 1} = f_a$.
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5)$, $\mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4)$, $\mu_{2\to c} = f_b$, $\mu_{1\to c} = f_a$.
3. $\mu_{3\to c} = (\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5))$, $\mu_{c\to 3} = \sum_{x_1,x_2} f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3)$.
4. $\mu_{c\to 1} = \sum_{\sim x_1} \mu_{2\to c}\mu_{3\to c} f_c$, $\mu_{c\to 2} = \sum_{\sim x_2} \mu_{1\to c}\mu_{3\to c} f_c$, $\mu_{3\to d} = \mu_{c\to 3}\mu_{e\to 3}$, $\mu_{3\to e} = \mu_{c\to 3}\mu_{d\to 3}$.

Parallelization: An Example



1. $\mu_{5\to e} = 1, \mu_{4\to d} = 1, \mu_{b\to 2} = f_b, \mu_{a\to 1} = f_a$.
2. $\mu_{e\to 3} = \sum_{\sim x_3} f_e \cdot 1 = \sum_{x_5} f_e(x_3, x_5)$, $\mu_{d\to 3} = \sum_{x_4} f_d(x_3, x_4)$, $\mu_{2\to c} = f_b$, $\mu_{1\to c} = f_a$.
3. $\mu_{3\to c} = (\sum_{x_5} f_e(x_3, x_5))(\sum_{x_4} f_d(x_4, x_5))$, $\mu_{c\to 3} = \sum_{x_1,x_2} f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3)$.
4. $\mu_{c\to 1} = \sum_{\sim x_1} \mu_{2\to c}\mu_{3\to c} f_c$, $\mu_{c\to 2} = \sum_{\sim x_2} \mu_{1\to c}\mu_{3\to c} f_c$, $\mu_{3\to d} = \mu_{c\to 3}\mu_{e\to 3}$, $\mu_{3\to e} = \mu_{c\to 3}\mu_{d\to 3}$.
5. $\mu_{d\to 4} = \sum_{\sim x_4} \mu_{3\to d} f_d$, $\mu_{e\to 5} = \sum_{\sim x_5} \mu_{3\to e} f_e$.
6. Product of messages received by x_5:

$$\sum_{\sim x_5} f_e(x_3, x_5)(\sum_{x_4} f_d(x_3, x_4))(\sum_{x_1,x_2} f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3))$$

Generalized State Realization

- Three kinds of nodes: variables, constraints, and states.
- Constraints involve both variables and states.
- A configuration $(z_1, \ldots, z_n, s_1, \ldots, s_m)$, where z_i are variables, and s_i are states, will be called admissible if satisfies all the constraints.
- A configuration of symbol variables only $(z_1, \ldots, z_n)$ will be called admissible if it satisfies all the constraint for some state assignment $(s_1, \ldots, s_m)$.
- Let $\mathbf{z} = (z_1, \ldots, z_n)$ be an admissible configuration. We assume that we have some weight function which factorizes as $w(z) = \prod_j w^j(z_j)$ mapping admissible configurations to real numbers.
- We are interested in computing

$$w_k(s_k) = \sum_{\mathbf{z} \in C_k(s_k)} w(\mathbf{z})$$

where $C_k(s_k)$ is the set of all admissible configurations consistent with s_k.

Belief Propagation

- Normal graphs: symbol variables have degree 1, state variables have degree 2.
- Break configurations into "upstream" and "downstream" parts.
- Define $\mathbf{w}_{k\uparrow}(s_k)$ to be the sum of the weights of all upstream configurations consistent with s_k; and $\mathbf{w}_{k\downarrow}$ to be the sum of the weights of all the downstream configurations consistent with s_k.

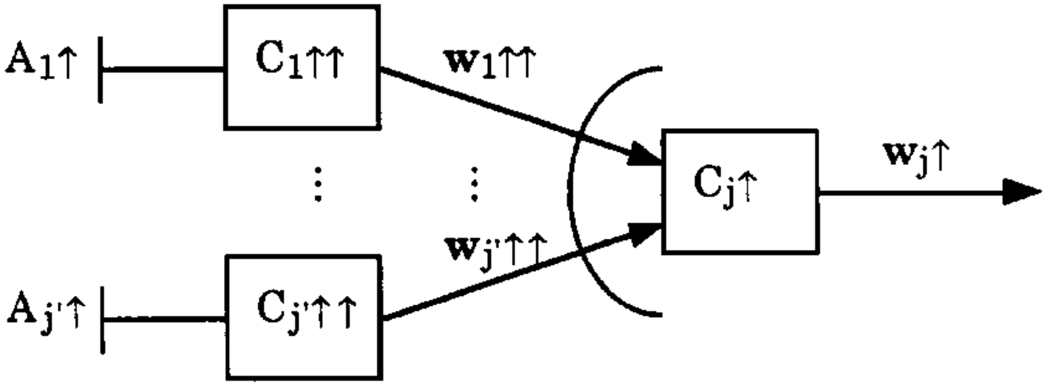


Figure 2: Note that state variable are shown as edges (without a dongle sign) in the picture.

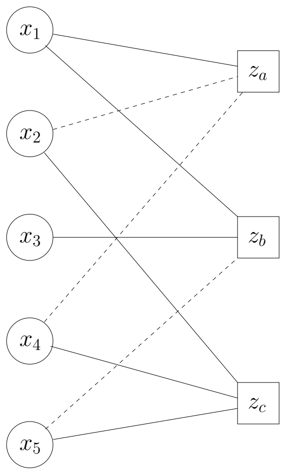
- The weight of an upstream configuration consistent with s_j is the product of the weights of subconfigurations along each branch $w_{k\uparrow\uparrow}$ in the picture:

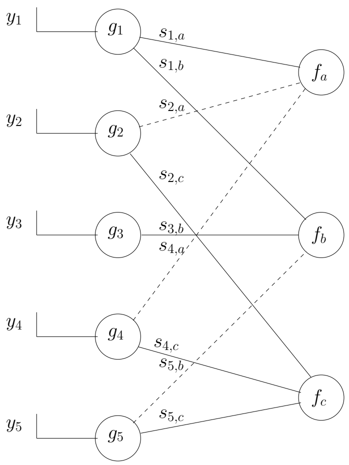
$$\mathbf{w}_{j\uparrow}(s_j) = \sum_{C_j(s_j)} \prod w_{k\uparrow\uparrow}(s_{j'}) \tag{3}$$

Generalized State Realization of Survey Propagation

1. Construct a normal realization of the K-sat problem as follows. Each variable i will be connected to a new constraint node g_i, and to no other nodes.

2. If before, the link (i, a) was in the factor graph, we add a state $s_{i,a}$ and connect g_i to $s_{i,a}$ and in turn $s_{i,a}$ to the constraint node a.





New Constraints

- Each variable node will take on a possible value of $\{0, 1, *\}$.
- Each state node will take a value from the alphabet $\{F, \bar{F}, *\} \times \{F, \bar{F}, *\}$.
- State will be written as $s_i = (l_i, r_i)$, where l_i will be called the left state, and r_i will be called the right state.
- Notation: $[X]$ be the indicator function of the event X, i.e. $[X] = 1$ if X is true, and $[X] = 0$ if X is not true.
- $u = v$ indicates that both u and v must take on the same value
- $u \sim v$ to indicates that either $u = v$ or $v = *$.
- $A_{i,a}$ will denote the map which maps $\{0, 1\}$ into $\{F, \bar{F}\}$ by outputing F if the input is the preferred assignment of constraint a for node i and $\bar{F}$ otherwise.

New Constraints

- Constraint associated with a constraint node a carried over from the original factor graph:

$$f_a = \prod_{s \in N(a)} \left([r(s) = F \text{ and all neighboring states } \hat{s} \neq s \text{ have } l(\hat{s}) = \bar{F}] + [r(s) = * \text{ and at least one state } \hat{s} \neq s \text{ has } l(\hat{s}) \neq \bar{F}] \right)$$

- Constraint associated with the new constraint nodes g_i:

$$g_i = \prod_{s \in N(g_i)} [A_{i,a}(y_i) \sim l(s_i)] \prod_{s_i \in N(g_i)} [l_i \sim r_i]$$

Weight of the solution

- A variable i is called constrained if for some a, $r(s_{i,a}) \in \{F, \bar{F}\}$, i.e. not all associated right-states are "*."
- A variable is called unconstrained if it is not constrained, but its value is not "*."
-

$$\begin{aligned} W_i &= \omega_* \text{ if } y_i = * \\ &= \omega_u \text{ if } y_i \text{ is unconstrained} \\ &= 1 \text{ if } y_i \text{ is constrained} \end{aligned}$$

- Define the weight of an assignment of values to state and variable nodes as

$$W = \prod_i W_i$$

  where the sum is taken over all variable and constraint nodes.
- For assignments of variables which satisfy all the constraints,

$$W = (\omega_u)^{\text{number of unconstrained variables}} (\omega_*)^{\text{number of star variables}}$$

Belief Propagation

- The state FF for example will appear in $s_{i,a}$ only if all the other state variables adjacent to a have $(l(s), r(s)) = (\bar{F}, *)$. Thus,

$$\rho_{a\to i}(FF) = \prod_{j\in N(a)-\{i\}} \lambda_{j\to a}(\bar{F}*)$$

-

$$\rho_{a\to i}(**) = \prod_{j\in N(a)-\{i\}} (\lambda_{j\to a}(F*) + \lambda_{j\to a}(\bar{F}*) + \lambda_{j\to a}(**)) - \prod_{j\in N(a)-\{i\}} \lambda_{j\to a}(\bar{F}*)$$

- The state $\bar{F}*$ will appear in $s_{i,a}$ if all the other nodes get $*$ right states but not all of them take the left state $\bar{F}$ (because that would imply we get an F right state). Alternatively, there may be a single FF state among the other nodes; but not more than one because that would violate constraint f_a. So:

$$\begin{aligned}\rho_{a\to i}(\bar{F}*) &= \prod_{j\in N(a)-\{i\}} (\lambda_{j\to a}(F*) + \lambda_{j\to a}(\bar{F}*) + \lambda_{j\to a}(**)) - \prod_{j\in N(a)-\{i\}} \lambda_{j\to a}(\bar{F}*) \\ &+ \sum_{k\in N(a)-\{i\}} (\lambda_{k\to a}(FF) - \lambda_{k\to a}(F*) - \lambda_{k\to a}(**)) \prod_{m\in N(a)-\{i,k\}} \lambda_{m\to a}(\bar{F}*)\end{aligned}$$

Belief Propagation

- State $s_{i,a}$ will take the value FF if it gets no Fs from those constraints which disagree with a and Fs or $*$s from those constraints which agree.

$$\lambda_{i\to a}(FF) = \prod_{v\in C_a^u(i)} \rho_{b\to i}(\bar{F}*) \prod_{v\in C_a^s(i)} (\rho_{b\to i}(FF) + \rho_{b\to i}(F*))$$

- The state $s_{i,a}$ will take the value $**$ only if all of its incoming values are $*$s. Since i is a star, this assignment has weight ω_*. Thus:

$$\lambda_{i\to a}(**) = \omega_* \prod_{b\in N(i)-\{a\}} \rho(b \to i)(**)$$

- The state $s_{i,a}$ will take the value $\bar{F}*$ if it receives only $*$s from those constraints which agree with a, and F and $*$s from those which do not. However, we must also take note that the assignment where i receives all $*$s and takes the value of $\bar{F}$ must be weighed by ω_u, because here i is an unconstrained variable. Thus:

$$\lambda_{i\to a}(\bar{F}*) = \prod_{b\in C_a^s(i)} \rho_{i\to a}(\bar{F}*)(\prod_{b\in C_a^u(i)} (\rho_{i\to a}(F*) + \rho_{i\to a}(FF)) - (1-\omega_u) \prod_{b\in C_a^u(i)} \rho_{i\to a}(F*))$$

Survey Propagation

Let $\omega_u + \omega_* = 1$, and let us initialize the algorithm with $\rho_{a\to i}(F*) = \rho_{a\to i}(**) = \rho_{a\to i}(\bar{F}*)$ and $\rho_{a\to i}(\bar{F}*) + \rho_{a\to i}(FF) = 1$ for all messages $\rho_{a\to i}$. Then, these relations persist throughout the execution of the algorithm, and

$$\rho_{a\to i}(FF) = \eta_{a\to i}$$

$$\lambda_{j\to a}(F*) + \lambda_{j\to a}(**) = \Pi^s_{j\to a} + \Pi^*_{j\to a}$$
$$\lambda_{i\to a}(\bar{F}*) = \Pi^u_{i\to a}$$

Proof that $\lambda_{i\to a}(\bar{F}*) = \Pi^u_{i\to a}$

$$\lambda_{i\to a}(\bar{F}*) = \prod_{b\in C^s_a(i)} \rho_{i\to a}(\bar{F}*)(\prod_{b\in C^u_a(i)} (\rho_{i\to a}(F*)+\rho_{i\to a}(FF))-(1-\omega_u)\prod_{b\in C^u_a(i)} \rho_{i\to a}(F*))$$

Because $\rho_{b\to i}(\bar{F}*) = 1 - \rho_{b\to i}(FF)$ and because $\rho_{b\to i}(F*) + \rho_{b\to i}(FF) = 1$, we have

$$\lambda_{i\to a}(\bar{F}*) = \prod_{b\in C^s_a(i)} (1-\rho_{i\to a}(FF))(\prod_{b\in C^u_a(i)} 1-(1-\omega_u)\prod_{b\in C^u_a(i)} (1-\rho_{i\to a}(FF))$$

And using the fact that $\rho_{v\to i}(FF) = \eta_{v\to i}$, we have the survey propagation update rule with $\omega_u = 0$:

$$\Pi^u_{i\to a} = [\prod_{b\in C^s_a(i)} (1-\eta_{a\to i})][1-\prod_{b\in C^u_a(i)} (1-\eta_{a\to i})$$

Summary and Conclusions

- Warning propagation: effective for low-density random K-sat problems.
- Survey propagation: effective for random K-sat problems up to the threshold (?)
- Survey propagation is belief propagation on a space with don't-care symbols.
- It appears that for a problem with many different clusters, approaches based on decimation and cluster modeling do well.