On the Interpolation of Product-Based Message Passing Heuristics for SAT

Oliver Gableske[1]

[1]Institute of Theoretical Computer Science
Ulm University
Germany

oliver@gableske.net
https://www.gableske.net

SAT 2013, 11.07.2013

Outline

Goals

1. Provide better access to MP for the SAT community.
   - Provide a consistent notational frame to explain all currently available MP heuristics.
   - Explain the functioning of all these heuristics.
   - Explain their respective strengths and weaknesses.
   - Explain where they differ.
2. Extend our knowledge about MP.
   - Provide more general/flexible MP heuristics.
   - Integrate MP into a CDCL solver (used to initialize VSIDS and phase-saving).

Message Passing on a conceptual level (1)

- Message Passing (MP) is a class of algorithms
- H $\in$ MP can be understood as variable and value ordering heuristics in the context of SAT
- The main goal of H is to provide *biases* for all variables of a CNF F
- $\forall v \in \mathcal{V} : \beta_H(v) \in [-1.0, 1.0]$
- The biases can be used to guide search (CDCL or SLS)

Message Passing on a conceptual level (1)

- Message Passing (MP) is a class of algorithms
- H $\in$ MP can be understood as variable and value ordering heuristics in the context of SAT
- The main goal of H is to provide *biases* for all variables of a CNF F
- $\forall v \in \mathcal{V} : \beta_H(v) \in [-1.0, 1.0]$
- The biases can be used to guide search (CDCL or SLS)

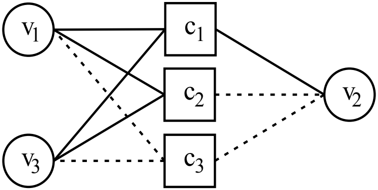
- Given the formula F, what does H do to compute the biases?

Message Passing on a conceptual level (2)

Example

$F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \bar{v_2} \lor v_3) \land (\bar{v_1} \lor \bar{v_2} \lor \bar{v_3})$

It is helpful to understand F as a *factor graph*.

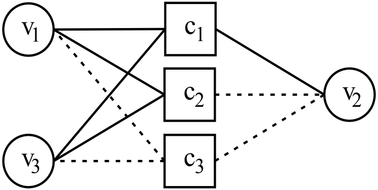


- Undirected, bipartite graph
- Two types of nodes (variable nodes (circles), clause nodes (squares))
- Two types of edges (positive edges (solid), negative edges (dashed))
- Edges constitute literal occurrences

Message Passing on a conceptual level (3)

Example

$F = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \bar{v_2} \vee v_3) \wedge (\bar{v_1} \vee \bar{v_2} \vee \bar{v_3})$



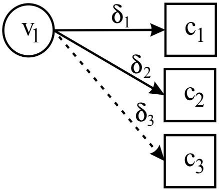
- H sends around messages along the edges.
- Assume variable v is contained in clause c as literal l

Two types of messages.

Message Passing on a conceptual level (4)

Example

$F = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \bar{v_2} \vee v_3) \wedge (\bar{v_1} \vee \bar{v_2} \vee \bar{v_3})$



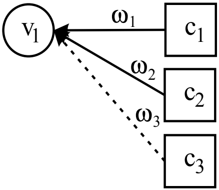
1. Disrespect Messages (from variable nodes towards clause nodes):
   - $\delta_H(l, c) \in [0.0, 1.0]$
   - The chance that l will *not* satisfy c

Intuitive meaning of $\delta_H(l, c) \approx 1.0$:
Variable v tells clause c that it cannot satisfy it.

Message Passing on a conceptual level (5)

Example

$F = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \bar{v_2} \vee v_3) \wedge (\bar{v_1} \vee \bar{v_2} \vee \bar{v_3})$



2. Warning Messages (from clause nodes towards variable nodes):
   - $\omega_H(c, v) \in [0.0, 1.0]$
   - The chance that *no other* literal in c can satisfy c

Intuitive meaning of $\omega_H(c, v) \approx 1.0$:
Clause c is telling variable v, that it needs it to be satisfied.

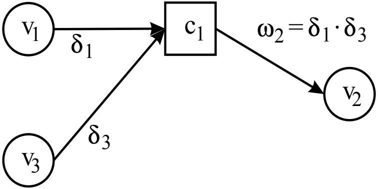
Message Passing on a conceptual level (6)

Example

$F = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \bar{v_2} \vee v_3) \wedge (\bar{v_1} \vee \bar{v_2} \vee \bar{v_3})$

For all product-based MP heuristics, the waring message is computed by

$$\omega_{\mathsf{H}}(c, v) = \prod_{l \in c \setminus \{v, \bar{v}\}} \delta_{\mathsf{H}}(l, c)$$



Message Passing on a conceptual level (7)

For all product-based MP heuristics, the *cavity freedom values* are computed by

$$[0.0, 1.0] \ni S_{\mathsf{H}}(l, c) = \begin{cases} \prod_{d \in C_v^-} [1 - \omega_{\mathsf{H}}(d, v)], l = v \\ \prod_{d \in C_v^+} [1 - \omega_{\mathsf{H}}(d, v)], l = \bar{v} \end{cases}$$

Intuitive meaning:
How happy are the other clauses if l satisfies c?

$$[0.0, 1.0] \ni U_{\mathsf{H}}(l, c) = \begin{cases} \prod_{d \in C_v^+ \setminus \{c\}} [1 - \omega_{\mathsf{H}}(d, v)], l = v \\ \prod_{d \in C_v^- \setminus \{c\}} [1 - \omega_{\mathsf{H}}(d, v)], l = \bar{v} \end{cases}$$

Intuitive meaning:
How happy are the other clauses if l does not satisfy c?

Message Passing on a conceptual level (8)

In summary:

- Computed δ_H values allow us to compute the ω_H values
- Computed ω_H values allow us to compute the $S_{\text{H}}, U_{\text{H}}$ values

However:

- H will not send around messages arbitrarily
- H performs *clause updates* $\forall c \in F$
- The ordering of the clauses in which they receive updates is determined by a *random clause permutation* $\pi \in \mathcal{S}_m$

Message Passing on a conceptual level (9)

Following $\pi \in \mathcal{S}_m$, each clause is updated exactly once.

Basically, a clause update for clause c consists of three steps.

1. Compute $\forall l \in c : \delta_{\mathsf{H}}(l, c)$
2. Using the δ, compute $\forall v \in c : \omega_{\mathsf{H}}(c, v)$
3. Using the ω, compute $\forall l \in c : S_{\mathsf{H}}(l, c), U_{\mathsf{H}}(l, c)$

Where do the δ values come from in order to compute a clause update?

Message Passing on a conceptual level (10)

We need the terms of *iteration* and *cycle* to explain that.

- Doing the clause updates for all clauses exactly once is called an *iteration*.
- A *cycle* is a finite tuple of iterations.
- Iterations and cycles capture the notion of *passing time* while H performs its computations.
- An iteration is a single point in time, a cycle is a time-frame.

We denote the specific values computed in iteration z of cycle y with

- ${}^y_z\delta_{\mathsf{H}}(l, c)$
- ${}^y_z\omega_{\mathsf{H}}(c, v)$
- ${}^y_zS_{\mathsf{H}}(l, c)$
- ${}^y_zU_{\mathsf{H}}(l, c)$

Message Passing on a conceptual level (11)

Again, in order to compute the clause update for iteration z in cycle y

1. Compute $\forall l \in c : {}^y_z\delta_{\mathsf{H}}(l,c)$
2. Using the ${}^y_z\delta_{\mathsf{H}}(l,c)$, compute $\forall v \in c : {}^y_z\omega_{\mathsf{H}}(c,v)$
3. Using the ${}^y_z\omega_{\mathsf{H}}(c,v)$, compute $\forall l \in c : {}^y_zS_{\mathsf{H}}(l,c), {}^y_zU_{\mathsf{H}}(l,c)$

Again, where do the ${}^y_z\delta_{\mathsf{H}}(l,c)$ values come from in order to compute a clause update?

Message Passing on a conceptual level (12)

The initialization for cycle y happens in iteration $z = 0$.

- $\forall c \in F : \forall l \in c$: initialize randomly with ${}^{y}_{0}\delta_{\mathsf{H}}(l,c) \in_R (0.0, 1.0)$
- The values for ${}^{y}_{0}\omega_{\mathsf{H}}(c,v), {}^{y}_{0}S_{\mathsf{H}}(l,c), {}^{y}_{0}U_{\mathsf{H}}(l,c)$ then directly follow with the definitions.

The clause updates for cycle y and iteration $z > 0$ are defined recursive.

- Rely on ${}^{\;\;y}_{z-1}S_{\mathsf{H}}(l,c), {}^{\;\;y}_{z-1}U_{\mathsf{H}}(l,c)$ in order to compute ${}^{y}_{z}\delta_{\mathsf{H}}(l,c)$.

How exactly is ${}^{y}_{z}\delta_{\mathsf{H}}(l,c)$ computed using these values?

Message Passing on a conceptual level (12)

The initialization for cycle y happens in iteration $z = 0$.

- $\forall c \in F : \forall l \in c$: initialize randomly with ${}^{y}_{0}\delta_{\mathsf{H}}(l, c) \in_R (0.0, 1.0)$
- The values for ${}^{y}_{0}\omega_{\mathsf{H}}(c, v), {}^{y}_{0}S_{\mathsf{H}}(l, c), {}^{y}_{0}U_{\mathsf{H}}(l, c)$ then directly follow with the definitions.

The clause updates for cycle y and iteration $z > 0$ are defined recursive.

- Rely on ${}^{y}_{z-1}S_{\mathsf{H}}(l, c), {}^{y}_{z-1}U_{\mathsf{H}}(l, c)$ in order to compute ${}^{y}_{z}\delta_{\mathsf{H}}(l, c)$.

How exactly is ${}^{y}_{z}\delta_{\mathsf{H}}(l, c)$ computed using these values?

- *This must be defined by H!*

Message Passing on a conceptual level (13)

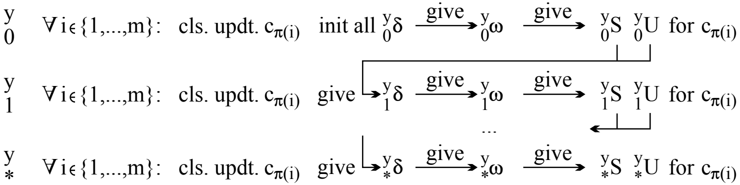
For Belief Propagation (BP) this is defined as

- $${}^{y}_{z}\delta_{\mathsf{BP}}(l,c) = \frac{{}^{\;\,y}_{z-1}U_{\mathsf{BP}}(l,c)}{{}^{\;\,y}_{z-1}U_{\mathsf{BP}}(l,c) + {}^{\;\,y}_{z-1}S_{\mathsf{BP}}(l,c)} \left(= \frac{U}{U+S}\right)$$

Message Passing on a conceptual level (14)

We now know

- . . . how cycles start.
- . . . how the iterations are done.

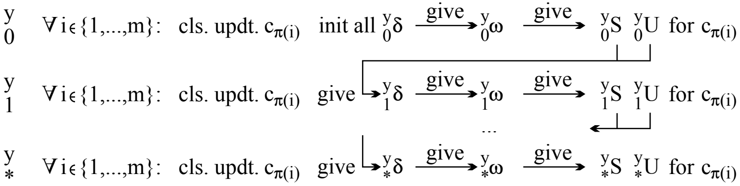


We do not know

Message Passing on a conceptual level (14)

We now know

- . . . how cycles start.
- . . . how the iterations are done.



We do not know

- . . . how a cycle terminates.
- What we need is an *abort condition*.

Message Passing on a conceptual level (15)

The abort conditions for a product-based MP heuristics is defined as

- $\forall c \in F : \forall v \in c : |\,{}^{y}_{z}\omega_{\mathsf{H}}(c,v) - {}^{\;\;\,y}_{z-1}\omega_{\mathsf{H}}(c,v)| < \omega_{\mathsf{max}}$
- In practice $\omega_{\mathsf{max}} = 0.01$

The iteration of cycle y in which the abort condition holds is denoted $*$. The messages

- ${}^{y}_{*}\delta_{\mathsf{H}}(l,c)$
- ${}^{y}_{*}\omega_{\mathsf{H}}(c,v)$

are called *equilibrium messages*.

The ${}^{y}_{*}\omega_{\mathsf{H}}(c,v)$ are used to compute the biases for cycle y.

Message Passing on a conceptual level (16)

Computing biases is done in three steps using the ${}^{y}_{*}\omega_{\mathsf{H}}(c,v)$.

1. Compute the *variable freedom* to be assigned to true ($\mathcal{T}$) or false ($\mathcal{F}$)

$$ {}^{y}\mathcal{T}_{\mathsf{H}}(v) = \prod_{c \in C_v^-} \left[1 - {}^{y}_{*}\omega_{\mathsf{H}}(c,v)\right] \quad {}^{y}\mathcal{F}_{\mathsf{H}}(v) = \prod_{c \in C_v^+} \left[1 - {}^{y}_{*}\omega_{\mathsf{H}}(c,v)\right] $$

2. Compute *magnetization values* using $\mathcal{T}$ and $\mathcal{F}$

$$ {}^{y}\mu_{\mathsf{H}}^{+}(v), {}^{y}\mu_{\mathsf{H}}^{-}(v), {}^{y}\mu_{\mathsf{H}}^{\pm}(v) \in [0.0, 1.0] $$

   These give ${}^{y}\mu_{\mathsf{H}}(v) = {}^{y}\mu_{\mathsf{H}}^{+}(v) + {}^{y}\mu_{\mathsf{H}}^{-}(v) + {}^{y}\mu_{\mathsf{H}}^{\pm}(v)$

3. Compute the biases

$$ {}^{y}\beta_{\mathsf{H}}^{+}(v) = \frac{{}^{y}\mu_{\mathsf{H}}^{+}(v)}{{}^{y}\mu_{\mathsf{H}}(v)} \quad {}^{y}\beta_{\mathsf{H}}^{-}(v) = \frac{{}^{y}\mu_{\mathsf{H}}^{-}(v)}{{}^{y}\mu_{\mathsf{H}}(v)} \quad {}^{y}\beta_{\mathsf{H}}(v) = {}^{y}\beta_{\mathsf{H}}^{+}(v) - {}^{y}\beta_{\mathsf{H}}^{-}(v) $$

Message Passing on a conceptual level (17)

Where do the ${}^{y}\mu^{+}_{\mathsf{H}}(v), {}^{y}\mu^{-}_{\mathsf{H}}(v), {}^{y}\mu^{\pm}_{\mathsf{H}}(v) \in [0.0, 1.0]$ come from?
Again, this must be defined by H!

For Belief Propagation (BP), this is defined as

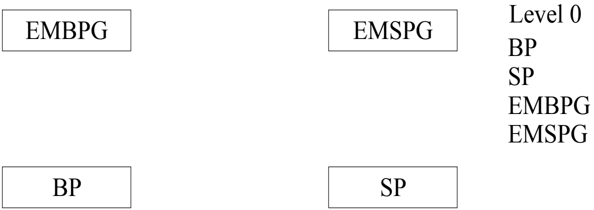
- ${}^{y}\mu^{+}_{\mathsf{BP}}(v) = {}^{y}\mathcal{T}_{\mathsf{BP}}(v)$
- ${}^{y}\mu^{-}_{\mathsf{BP}}(v) = {}^{y}\mathcal{F}_{\mathsf{BP}}(v)$
- ${}^{y}\mu^{\pm}_{\mathsf{BP}}(v) = 0$

Therefore, ${}^{y}\mu_{\mathsf{BP}}(v) = {}^{y}\mathcal{T}_{\mathsf{BP}}(v) + {}^{y}\mathcal{F}_{\mathsf{BP}}(v)$.

Finally, for BP, it is ${}^{y}\beta_{\mathsf{BP}}(v) = \dfrac{{}^{y}\mathcal{T}_{\mathsf{BP}}(v) - {}^{y}\mathcal{F}_{\mathsf{BP}}(v)}{{}^{y}\mathcal{T}_{\mathsf{BP}}(v) + {}^{y}\mathcal{F}_{\mathsf{BP}}(v)}$

Product-based MP heuristics (1)

Well known product-based MP heuristics.



- All the basic MP heuristics have different strengths and weaknesses.
- Introducing MP into a solver to guide its search is problematic.
- The necessity to choose basically means: However you choose, you choose wrong!

Product-based MP heuristics (2)

Increase the flexibility of MP heuristics in order to overcome the "robustness problem".

How to create a more flexible MP heuristic?

Product-based MP heuristics (2)

Increase the flexibility of MP heuristics in order to overcome the "robustness problem".

How to create a more flexible MP heuristic?

- Interpolation!

Interpolation (1)

What is it, that needs to be achieved in order to create an interpolation?
Given two product-based MP heuristics H_1 and H_2, we want an interpolation $\rho\mathsf{H}^i$, s.t.

- interpolation parameter $\rho \in [0.0, 1.0]$
- Setting $\rho = 0$ will make $\rho\mathsf{H}^i$ mimic H_1, i.e. $\beta_{\mathsf{H}_1}(v) = \beta^i_{\rho\mathsf{H}}(v, 0)$
- Setting $\rho = 1$ will make $\rho\mathsf{H}^i$ mimic H_2, i.e. $\beta_{\mathsf{H}_2}(v) = \beta^i_{\rho\mathsf{H}}(v, 1)$
- Setting $\rho \in (0.0, 1.0)$ results in a gradual adaption between H_1, H_2
  - gradually adapt the convergence behavior
  - gradually adapt the carefulness to present biases

Interpolation (2)

Equations used in all product-based MP heuristics.

During Iterations

- Disrespect message ${}^{y}_{z}\delta_{\mathsf{H}}(l,c)$
- Warning message ${}^{y}_{z}\omega_{\mathsf{H}}(l,c)$
- Literal cavity freedom values ${}^{y}_{z}S_{\mathsf{H}}(l,c), {}^{y}_{z}U_{\mathsf{H}}(l,c)$

After convergence, provided ${}^{y}_{*}\omega_{\mathsf{H}}(l,c)$

- Variable freedom ${}^{y}\mathcal{T}_{\mathsf{H}}(v), {}^{y}\mathcal{F}_{\mathsf{H}}(v)$
- Variable magnetization ${}^{y}\mu^{+}_{\mathsf{H}}(v), {}^{y}\mu^{-}_{\mathsf{H}}(v), {}^{y}\mu^{\pm}_{\mathsf{H}}(v), {}^{y}\mu_{\mathsf{H}}(v)$
- Variable bias ${}^{y}\beta^{+}_{\mathsf{H}}(v), {}^{y}\beta^{-}_{\mathsf{H}}(v), {}^{y}\beta_{\mathsf{H}}(v)$

Interpolation (3)

Equations that must be defined by H itself.

During Iterations

- Disrespect message ${}^{y}_{z}\delta_{\mathsf{H}}(l, c)$
- Warning message ${}^{y}_{z}\omega_{\mathsf{H}}(l, c)$
- Literal cavity freedom values ${}^{y}_{z}S_{\mathsf{H}}(l, c), {}^{y}_{z}U_{\mathsf{H}}(l, c)$

After convergence, provided ${}^{y}_{*}\omega_{\mathsf{H}}(l, c)$

- Variable freedom ${}^{y}\mathcal{T}_{\mathsf{H}}(v), {}^{y}\mathcal{F}_{\mathsf{H}}(v)$
- Variable magnetization ${}^{y}\mu^{+}_{\mathsf{H}}(v), {}^{y}\mu^{-}_{\mathsf{H}}(v), {}^{y}\mu^{\pm}_{\mathsf{H}}(v)$, ${}^{y}\mu_{\mathsf{H}}(v)$
- Variable bias ${}^{y}\beta^{+}_{\mathsf{H}}(v), {}^{y}\beta^{-}_{\mathsf{H}}(v), {}^{y}\beta_{\mathsf{H}}(v)$

Must be defined for the interpolation.

ISI (1)

ISI is a technique to derive $\rho\mathsf{H}^i$ given H_1 and H_2.
It uses an interpolation parameter $\rho \in [0.0, 1.0]$.
It derives

- ${}^y_z\delta^i_{\rho\mathsf{H}}(l, c, \rho), {}^y\mu^{i+}_{\rho\mathsf{H}}(v, \rho), {}^y\mu^{i-}_{\rho\mathsf{H}}(v, \rho), {}^y\mu^{i\pm}_{\rho\mathsf{H}}(v, \rho)$

given

- ${}^y_z\delta_{\mathsf{H}_1}(l, c), {}^y\mu^{+}_{\mathsf{H}_1}(v), {}^y\mu^{-}_{\mathsf{H}_1}(v), {}^y\mu^{\pm}_{\mathsf{H}_1}(v)$
- ${}^y_z\delta_{\mathsf{H}_2}(l, c), {}^y\mu^{+}_{\mathsf{H}_2}(v), {}^y\mu^{-}_{\mathsf{H}_2}(v), {}^y\mu^{\pm}_{\mathsf{H}_2}(v)$

How exatly does it work? Exemplary explanation.
Assume we want to

- interpolate BP and SP
- using interpolation parameter $\rho \in [0.0, 1.0]$
- in order to derive the interpolation $\rho\mathsf{SP}^i$

ISI (2)

Step 1. derives ${}^y_z\delta^i_{\rho\mathsf{SP}}(l, c, \rho)$ using

- ${}^y_z\delta_{\mathsf{BP}}(l, c) = \frac{U}{U+S}$
- ${}^y_z\delta_{\mathsf{SP}}(l, c) = \frac{U(1-S)}{U(1-S)+S}$

Linearly interpolate!

Numerator:

$(1-\rho)\{U\} + \rho\{U(1-S)\} = \ldots = U(1-\rho S)$

Denominator:

$(1-\rho)\{U+S\} + \rho\{U(1-S)+S\} = \ldots = U(1-\rho S) + S$

Combine:

$${}^y_z\delta^i_{\rho\mathsf{SP}}(l, c, \rho) = \frac{U(1-\rho S)}{U(1-\rho S)+S}$$

ISI (3)

Step 2. derives ${}^y\mu^{i+}_{\rho\mathsf{SP}}(v,\rho)$ using

- ${}^y\mu^{+}_{\mathsf{BP}}(v) = {}^y\mathcal{T}_{\mathsf{BP}}(v)$ $\qquad (= \mathcal{T})$
- ${}^y\mu^{+}_{\mathsf{SP}}(v) = {}^y\mathcal{T}_{\mathsf{SP}}(v)(1 - {}^y\mathcal{F}_{\mathsf{SP}}(v))$ $\qquad (= \mathcal{T}(1-\mathcal{F}))$

Linearly interpolate!

$(1-\rho)\{\mathcal{T}\} + \rho\{\mathcal{T}(1-\mathcal{F})\} = \ldots = \mathcal{T}(1-\rho\mathcal{F}) = {}^y\mu^{i+}_{\rho\mathsf{SP}}(v,\rho)$

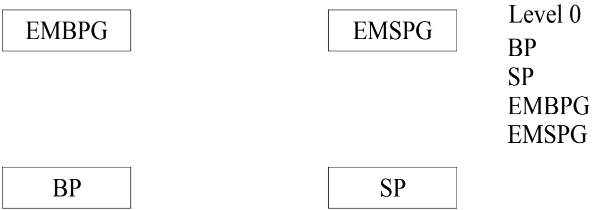
Step 3. derives ${}^y\mu^{i-}_{\rho\mathsf{SP}}(v,\rho)$ in a similar way.

Step 4. derives ${}^y\mu^{i\pm}_{\rho\mathsf{SP}}(v,\rho)$ in a similar way.

In the end, all four defining functions for $\rho\mathsf{SP}^i$ have been derived.

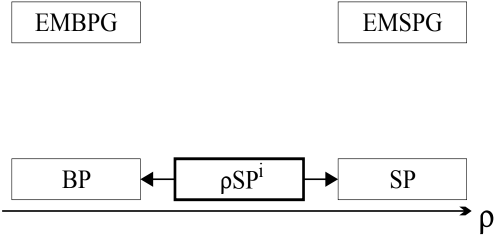
The product-based MP Hierarchy (1)

The basic product-based MP heuristics.



The product-based MP Hierarchy (2)

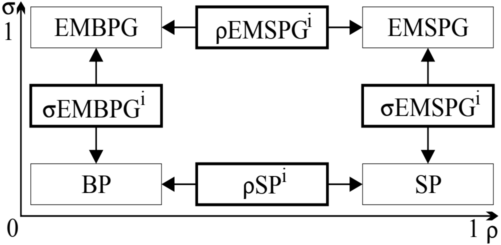
The first level of interpolations (applying ISI once).



Level 0	Level 1
BP	ρSP^i
SP	
EMBPG	
EMSPG	

The product-based MP Hierarchy (3)

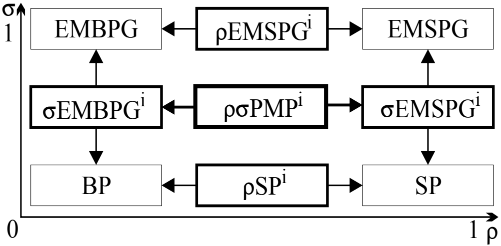
The first level of interpolations (applying ISI once).



Level 0	Level 1
BP	σEMBPG^i
SP	σEMSPG^i
EMBPG	ρSP^i
EMSPG	ρEMSPG^i

The product-based MP Hierarchy (4)

The first level of interpolations (applying ISI once).



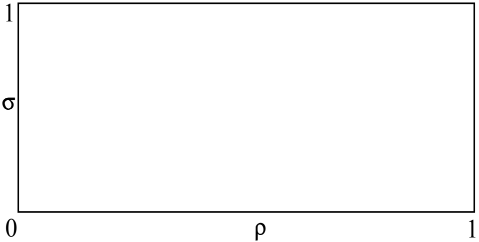
Level 0	Level 1	Level 2
BP	σEMBPG^i	ρσPMP^i
SP	σEMSPG^i	
EMBPG	ρSP^i	
EMSPG	ρEMSPG^i	

$\rho\sigma$PMP^i (1)

Why is $\rho\sigma$PMP^i so special?

- It is the most general product-based MP heuristic.
- It can mimic the behavior of all others.
- It can provide MP behavior that cannot be achieved by any other heuristic.

Each point in the parameter plane $(\rho, \sigma) \in [0.0, 1.0]^2$ characterizes a specific MP behavior.

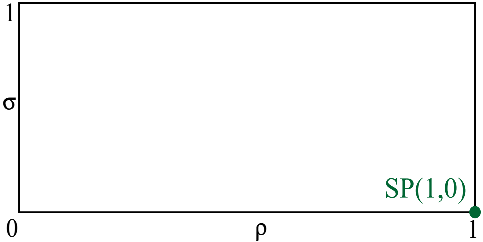


$\rho\sigma$PMP^i (2)

Why is $\rho\sigma$PMP^i so special?

- It is the most general product-based MP heuristic.
- It can mimic the behavior of all others.
- It can provide MP behavior that cannot be achieved by any other heuristic.

Each point in the parameter plane $(\rho, \sigma) \in [0.0, 1.0]^2$ characterizes a specific MP behavior.



Best behavior given?

Can use:

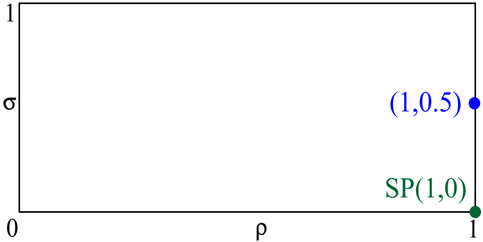
SP, ρSP^i, σEMSPG^i, ρσPMP^i

$\rho\sigma$PMP^i (3)

Why is $\rho\sigma$PMP^i so special?

- It is the most general product-based MP heuristic.
- It can mimic the behavior of all others.
- It can provide MP behavior that cannot be achieved by any other heuristic.

Each point in the parameter plane $(\rho, \sigma) \in [0.0, 1.0]^2$ characterizes a specific MP behavior.



Best behavior given?

Can use:
SP, ρSP^i, σEMSPG^i, $\rho\sigma$PMP^i

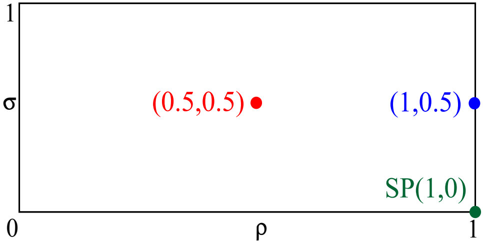
Can use:
σEMSPG^i, $\rho\sigma$PMP^i

$\rho\sigma$PMP^i (4)

Why is $\rho\sigma$PMP^i so special?

- It is the most general product-based MP heuristic.
- It can mimic the behavior of all others.
- It can provide MP behavior that cannot be achieved by any other heuristic.

Each point in the parameter plane $(\rho, \sigma) \in [0.0, 1.0]^2$ characterizes a specific MP behavior.



Best behavior given?

Can use:
SP, ρSP^i, σEMSPG^i, ρσPMP^i

Can use:
σEMSPG^i, ρσPMP^i

Can use:
ρσPMP^i

$\rho\sigma$PMP^i (5)

Why is that good in order to introduce MP into a solver?

- This circumvents the need to choose from all the available MP heuristics.
- The interpolation parameters ρ, σ can be tuned automatically for each class of formulas.

In the context of a CDCL search:

1. Use $\rho\sigma$PMP^i to compute biases.
2. Use a specifically tuned MP behavior for the formula class.
3. Use the biases to initialize VSIDS and phase-saving.

Empirical results from parameter tuning

Benchmark	S/U	Solver Performance					
		DimetheusJW		DimetheusMP			
		%	PAR10	%	PAR10	ρ	σ
battleship	S	47.4	10627.2	**89.5**	**2130.1**	0.5002	0.0025
battleship	U	55.6	8919.7	55.6	**8890.4**	0.4463	1.0000
em-all	S	75.0	5263.7	**100.0**	**75.4**	0.8606	0.1295
em-compact	S	0.0	20000.0	**37.5**	**12728.5**	0.9229	0.7946
em-explicit	S	75.0	5473.3	**100.0**	**157.1**	0.2932	0.2698
em-fbcolors	S	12.5	17723.3	**37.5**	**12662.9**	0.0000	0.1731
grid-pebbling	S	100.0	16.5	100.0	**8.0**	0.9931	0.3890
grid-pebbling	U	88.9	2226.9	**100.0**	**4.7**	0.5884	0.0035
sgen1	S	16.7	16677.7	**27.8**	**14460.9**	0.0937	0.6563
k3-r4.200	S	0.0	20000.0	**100.0**	**22.7**	0.9929	0.0004
k3-r4.237	S	0.0	20000.0	**75.0**	**5026.8**	0.9961	0.0000
k4-r9.000	S	0.0	20000.0	**100.0**	**10.0**	0.8592	0.0000
k4-r9.526	S	0.0	20000.0	**100.0**	**5.2**	0.9530	0.0000

Conclusions

1. Provided better access to MP for the SAT community.
   - We provided a unified and consistent notational frame to explain all currently available MP heuristics.
   - We explained the functioning of all these heuristics.
   - We explained their respective strengths and weaknesses.
   - We explained where they differ.
2. Extend our knowledge about MP.
   - We provided a hierarchy of generality regarding product-based MP heuristics.
   - We clarified what an interpolation is and how they are derived.
   - Integrated MP into a CDCL solver (used to initialize VSIDS and phase-saving) to get more empirical insight.

Thanks you for your attention!

You can send disrespect messages and questions to
oliver@gableske.net

Thank you for your attention.

Bibliography

Check the paper

O. Gableske

On the Interpolation between Product-Based Message Passing Heuristics for SAT

published in

Theory and Application of Satisfiability Testing – SAT 2013
LNCS 7962, pp. 293–308. Springer, Heidelberg, 2013

The difference between BP and SP

With $\rho, S, U, \mathcal{T}, \mathcal{F} \in [0.0, 1.0]$

Disrespect messages:

- ${}^{y}_{z}\delta_{\mathsf{BP}}(l,c) = \dfrac{U}{U+S}$ $\qquad {}^{y}_{z}\delta_{\mathsf{SP}}(l,c) = \dfrac{U(1-S)}{U(1-S)+S}$
- ${}^{y}_{z}\delta^{i}_{\rho\mathsf{SP}}(l,c,\rho) = \dfrac{U(1-\rho S)}{U(1-\rho S)+S}$

Bias computations:

- ${}^{y}\beta_{\mathsf{BP}}(v) = \dfrac{\mathcal{T}-\mathcal{F}}{\mathcal{T}+\mathcal{F}}$ $\qquad {}^{y}\beta_{\mathsf{SP}}(v) = \dfrac{\mathcal{T}-\mathcal{F}}{\mathcal{T}+\mathcal{F}-\mathcal{T}\mathcal{F}}$
- ${}^{y}\beta^{i}_{\rho\mathsf{SP}}(v,\rho) = \dfrac{\mathcal{T}-\mathcal{F}}{\mathcal{T}+\mathcal{F}-\rho\mathcal{T}\mathcal{F}}$