SAT Solving with Message Passing

Oliver Gableske
Institute of Theoretical Computer Science
Ulm University

Doctoral thesis defense
December 19th, 2016
The SAT Problem

- we are dealing with propositional statements called **formulas**
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either **satisfiable** or **unsatisfiable**
The SAT Problem

- we are dealing with propositional statements called formulas
- \[ F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \]
- a formula is either satisfiable or unsatisfiable
  - find an assignment for each of the **Boolean variables** \( v_1, v_2, v_3 \)
  - to either \( v_i \mapsto 1 \) (true) or \( v_i \mapsto 0 \) (false)
  - such that the formula, under these assignments, evaluates to 1 (true)
we are dealing with propositional statements called formulas

\[ F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \]

a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{ v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0 \} \).
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \).

1. Replace the variables in \( F \) as indicated by \( \alpha \).
   \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{ v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0 \} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)

2. Evaluate the \( \neg \) (negation, ”not”) operators.
   \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{ v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0 \} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 1 \lor 0) \land (1 \lor 1 \lor 0) \)

2. Evaluate the \( \neg \) (negation, ”not”) operators.
   \( \alpha(F) = (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)

- a literal is either a Boolean variable \( l = v_i \) or its negation \( l = \neg v_i \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)
2. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)
3. Evaluate the \( \lor \) (disjunction, ”or”) operators.
   \( \alpha(F) = 1 \land 1 \land 1 \)

- a literal is either a Boolean variable \( l = v_i \) or its negation \( l = \neg v_i \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{ v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0 \} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)

2. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)

3. Evaluate the \( \lor \) \( \text{(disjunction, ”or”)} \) operators.
   \( \alpha(F) = 1 \land 1 \land 1 \)

- a \textbf{literal} is either a Boolean variable \( l = v_i \) or its negation \( l = \neg v_i \)
- a \textbf{clause} of size \( k \) is a disjunction of \( k \) literals, \( c = (l_1 \lor \ldots \lor l_k) \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{ v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0 \} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)
2. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)
3. \( \alpha(F) = 1 \land 1 \land 1 \)
4. Evaluate the \( \land \) (conjunction, ”and”) operators.
   \( \alpha(F) = 1 \)

- a **literal** is either a Boolean variable \( l = v_i \) or its negation \( l = \neg v_i \)
- a **clause** of size \( k \) is a disjunction of \( k \) literals, \( c = (l_1 \lor ... \lor l_k) \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)
2. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)
3. \( \alpha(F) = 1 \land 1 \land 1 \)

4. Evaluate the \( \land \) (conjunction, ”and” ) operators.
   \( \alpha(F) = 1 \)

- a literal is either a Boolean variable \( l = v_i \) or its negation \( l = \neg v_i \)
- a clause of size \( k \) is a disjunction of \( k \) literals, \( c = (l_1 \lor ... \lor l_k) \)
- a \( k \)-CNF formula is a conjunction of clauses of size \( \leq k \), \( F = c_1 \land c_2 \land ... \)
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \).

1. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor \neg 1 \lor 0) \land (\neg 1 \lor \neg 1 \lor \neg 0) \)
2. \( \alpha(F) = (1 \lor 1 \lor 0) \land (1 \lor 0 \lor 0) \land (0 \lor 0 \lor 1) \)
3. \( \alpha(F) = 1 \land 1 \land 1 \)
4. \( \alpha(F) = 1 \)

- we can rewrite any propositional statement into a logically equivalent formula in **conjunctive normal form (CNF)**
- we can therefore focus on formulas of this particular shape
The SAT Problem

- we are dealing with propositional statements called formulas
- $F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3)$
- a formula is either satisfiable or unsatisfiable

Example

Let $\alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\}$, then $\alpha(F) = 1$.
- Assignment $\alpha$ satisfies $F$, hence $F$ is **satisfiable**.
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \), then \( \alpha(F) = 1 \).
- Assignment \( \alpha \) satisfies \( F \), hence \( F \) is **satisfiable**.

Let \( \gamma = \{v_1 \mapsto 0, v_2 \mapsto 0, \mapsto 0\} \), then \( \gamma(F) = 0 \).
- Not all the possible assignments satisfy \( F \), although it is satisfiable.
The SAT Problem

- we are dealing with propositional statements called formulas
- \( F = (v_1 \lor v_2 \lor v_3) \land (v_1 \lor \neg v_2 \lor v_3) \land (\neg v_1 \lor \neg v_2 \lor \neg v_3) \)
- a formula is either satisfiable or unsatisfiable

Example

Let \( \alpha = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\} \), then \( \alpha(F) = 1 \).
- Assignment \( \alpha \) satisfies \( F \), hence \( F \) is **satisfiable**.

Let \( \gamma = \{v_1 \mapsto 0, v_2 \mapsto 0, \mapsto 0\} \), then \( \gamma(F) = 0 \).
- Not all the possible assignments satisfy \( F \), although it is satisfiable.

Example

\( G = (v_1) \land (\neg v_1) \) cannot be satisfied by any assignment.
- Formula \( G \) is **unsatisfiable**.
The SAT Problem

Definition (The $k$-SATISFIABILITY problem)

Given a $k$-CNF formula $F$. Determine whether $F$ is satisfiable.
The SAT Problem

Definition (The $k$-SATISFIABILITY problem)

Given a $k$-CNF formula $F$. Determine whether $F$ is satisfiable.

Theorem (Cook-Levin)

The $k$-SAT problem with $k > 2$ is $\text{NP}$-complete.$^1$

1. Solving the $k$-SAT problem is difficult.
2. The $k$-SAT problem is important.

$^1$Henceforth, we assume $k > 2$. 
Solving the $k$-SAT problem for a given $k$-CNF formula $F$ (i.e., deciding whether $F$ is satisfiable) can be done by constructing a satisfying assignment $\alpha$ for $F$.

- let computers do the work by running a SAT solver algorithm
SAT Solving

- Solving the $k$-SAT problem for a given $k$-CNF formula $F$ (i.e., deciding whether $F$ is satisfiable) can be done by constructing a satisfying assignment $\alpha$ for $F$.
- Let computers do the work by running a SAT solver algorithm.
- Decimate is a very simple SAT solver.
  - Starts from the empty assignment $\alpha = \{\}$ and successively extends it.
  - By making decisions on what variable to assign to what value.
Solving the \( k \)-SAT problem for a given \( k \)-CNF formula \( F \) (i.e., deciding whether \( F \) is satisfiable) can be done by constructing a satisfying assignment \( \alpha \) for \( F \).

- let computers do the work by running a **SAT solver** algorithm
- Decimate is a very simple SAT solver
  - starts from the empty assignment \( \alpha = \{\} \) and successively extends it
  - by making **decisions** on what variable to assign to what value
  - until either \( \alpha(F) = 0 \) (failure)

**Example**

\[
F = (\neg v_1 \lor v_2 \lor v_3) \land (\neg v_1 \lor v_2 \lor \neg v_3) \land (v_1 \lor v_2 \lor v_3) \land (v_1 \lor v_2 \lor \neg v_3)
\]

\[
\begin{array}{c}
\text{failure} \\
\alpha(F) = 0 \\
\text{output}
\end{array}
\]

"unknown"
SAT Solving

- Solving the $k$-SAT problem for a given $k$-CNF formula $F$ (i.e., deciding whether $F$ is satisfiable) can be done by constructing a satisfying assignment $\alpha$ for $F$.
- Let computers do the work by running a SAT solver algorithm.
- Decimate is a very simple SAT solver.
  - Starts from the empty assignment $\alpha = \{\}$ and successively extends it by making decisions on what variable to assign to what value.
  - Until either $\alpha(F) = 0$ (failure) or $\alpha(F) = 1$ (success).

Example

$F = (\neg v_1 \lor v_2 \lor v_3) \land (\neg v_1 \lor v_2 \lor \neg v_3) \land (v_1 \lor v_2 \lor v_3) \land (v_1 \lor v_2 \lor \neg v_3)$

![Diagram of the SAT solver process with variables $v_2$, $v_3$, and $v_1$ and their assignments 0 and 1, leading to failure or success with $\alpha(F) = 0$ or $\alpha(F) = 1$.]

Output:
- $\alpha(F) = 0$ (failure)
- $\alpha(F) = 1$ (success)

"Unknown" output when the result is not determined.
SAT Solving

- Solving the $k$-SAT problem for a given $k$-CNF formula $F$ (i.e., deciding whether $F$ is satisfiable) can be done by constructing a satisfying assignment $\alpha$ for $F$.

- Let computers do the work by running a SAT solver algorithm.

George Boole (* Nov. 2nd, 1815 – † Dec. 8th, 1864)

SAT solvers need to make clever decisions to be successful and fast!

Example

$$F = (\neg v_1 \lor v_2 \lor v_3) \land (\neg v_1 \lor v_2 \lor \neg v_3) \land (v_1 \lor v_2 \lor v_3) \land (v_1 \lor v_2 \lor \neg v_3)$$

- $\alpha(F) = 0$ (failure)
- $\alpha(F') = 1$ (success)
- Output: "unknown"
SAT Solving

- SAT solvers rely on **heuristics** to make the decisions
  - can be understood as “rules of thumb that can be calculated fast”
  - not perfect, they will not always lead to success
  - on average they should lead to success more often than to failure
- heuristics for SAT solving can be designed in many ways
  - one possibility is to rely on **Message Passing** algorithms
Message Passing (MP) Algorithms

- $F = (v_1 \lor \neg v_2) \land (\neg v_1 \lor v_2 \lor v_3) \land (v_2 \lor \neg v_3)$
- we can understand $F$ as a factor graph
Message Passing (MP) Algorithms

- $F = (v_1 \lor \neg v_2) \land (\neg v_1 \lor v_2 \lor v_3) \land (v_2 \lor \neg v_3)$
- send two types of messages along every edge

Diagram:

- Assume we pick $\alpha$ uniformly at random such that $\alpha(F) = 1$
  - $\delta(v, c) \in (0.0, 1.0)$, probability that $\alpha(v)$ does not satisfy $c$
  - $\omega(c, v) \in (0.0, 1.0)$, probability that $\alpha(c \setminus \{v\})$ is false
Message Passing (MP) Algorithms

- $F = (v_1 \lor \neg v_2) \land (\neg v_1 \lor v_2 \lor v_3) \land (v_2 \lor \neg v_3)$

- perform **iterative calculations of messages**

- randomly and independently initialize $\delta(l, c) \in (0.0, 1.0)$, then repeat
  - update the $\omega$ messages based on the $\delta$ messages
  - update the $\delta$ messages based on the $\omega$ messages
Message Passing (MP) Algorithms

- \[ F = (v_1 \lor \neg v_2) \land (\neg v_1 \lor v_2 \lor v_3) \land (v_2 \lor \neg v_3) \]

- **iterative calculations** are performed until **convergence**

- "no significant message-changes are observed anymore"

- yields the **equilibrium warning messages** \( \omega^* \)
Message Passing (MP) Algorithms

- the $\omega^*$ allow us to calculate variable marginals and biases
- Assume we select $\alpha$ uniformly at random from all assignments with $\alpha(F) = 1$. Then, we interpret the ...
  - **positive marginal** $\mu^+(v)$ as the probability to observe $\alpha(v) = 1$.
  - **negative marginal** $\mu^-(v)$ as the probability to observe $\alpha(v) = 0$.
  - **joker marginal** $\mu^*(v)$ as the probability to observe $\alpha(v) = v$.
- the **bias** $\beta(v) = \mu^+(v) - \mu^-(v) \in [-1.0, 1.0]$ reflects the predominant assignment to $v$ made by all assignments that satisfy $F$
- biases are not necessarily correct
Message Passing (MP) Algorithms

- the $\omega^*$ allow us to calculate variable marginals and biases
- Assume we select $\alpha$ uniformly at random from all assignments with $\alpha(F) = 1$. Then, we interpret the ...
  - **positive marginal** $\mu^+(v)$ as the probability to observe $\alpha(v) = 1$.
  - **negative marginal** $\mu^-(v)$ as the probability to observe $\alpha(v) = 0$.
  - **joker marginal** $\mu^*(v)$ as the probability to observe $\alpha(v) = v$.
- the **bias** $\beta(v) = \mu^+(v) - \mu^-(v) \in [-1.0, 1.0]$ reflects the predominant assignment to $v$ made by all assignments that satisfy $F$
- biases are not necessarily correct
- **MPDecide($\alpha, F$)** heuristic
  - assign the variable $v_i$ with strongest bias $|\beta(v_i)|$
  - assign as indicated by the sign of the bias
    - $\beta(v_i) < 0 \leadsto v_i \mapsto 0$
    - $\beta(v_i) > 0 \leadsto v_i \mapsto 1$
  - ties are broken at random
which leads to the MPDecimate solver
  - works just as Decimate
  - while relying on MPDecide(\(\alpha, F'\)) to make the decisions

\[
\begin{align*}
&v_2 \quad 0 \quad v_3 \quad 1 \quad v_1 \quad 0 \\
&v_2 \quad 1 \\
\end{align*}
\]

\(\alpha(F) = 0\) failure
\(\alpha(F') = 1\) success
output \(\alpha\)
output "unknown"
which leads to the MPDecimate solver

- works just as Decimate
- while relying on MPDecide($\alpha, F'$) to make the decisions

we want to empirically ascertain the quality of biases

that different MP algorithms MP$_1$ or MP$_2$ provide when

MP$_1$Decimate or MP$_2$Decimate must solve

a given set of formulas $\mathbb{F}$ called a benchmark

let MP$_1$/MP$_2$Decimate try to solve every $F \in \mathbb{F}$ exactly once

to calculate the success-rates

$S_1 = \#\text{success}_1/|\mathbb{F}|$ versus $S_2 = \#\text{success}_2/|\mathbb{F}|$
the literature provides two MP algorithms

- Belief Propagation (BP) \(\rightsquigarrow\) BPDecimate
- Survey Propagation (SP) \(\rightsquigarrow\) SPDecimate

used to solve uniform random 3-SAT formulas

- formulas are generated by generating clauses of size 3 at random
- the number of clauses is controlled by parameter \(r\) called ratio

generate 100 formulas for \(r\) to obtain benchmark \(F_r\)

determine the success-rates of BPDecimate and SPDecimate on \(F_r\)
the literature provides two MP algorithms
- Belief Propagation (BP) $\rightsquigarrow$ BPDecimate
- Survey Propagation (SP) $\rightsquigarrow$ SPDecimate

used to solve uniform random 3-SAT formulas
- formulas are generated by generating clauses of size 3 at random
- the number of clauses is controlled by parameter $r$ called ratio

generate 100 formulas for $r$ to obtain benchmark $\mathcal{F}_r$

determine the success-rates of BPDecimate and SPDecimate on $\mathcal{F}_r$
the literature provides two MP algorithms
  - Belief Propagation (BP) \(\rightsquigarrow\) BPDecimate
  - Survey Propagation (SP) \(\rightsquigarrow\) SPDecimate

used to solve uniform random 3-SAT formulas
  - formulas are generated by generating clauses of size 3 at random
  - the number of clauses is controlled by parameter \(r\) called ratio
  - generate 100 formulas for \(r\) to obtain benchmark \(F_r\)
  - determine the success-rates of BPDecimate and SPDecimate on \(F_r\)
SAT Solving with Message Passing

- The literature provides two MP algorithms:
  - Belief Propagation (BP) $\leadsto$ BPDecimate
  - Survey Propagation (SP) $\leadsto$ SPDecimate

- Used to solve **uniform random 3-SAT formulas**
  - Formulas are generated by generating clauses of size 3 at random.
  - The number of clauses is controlled by parameter $r$ called the ratio.
  - Generate 100 formulas for $r$ to obtain benchmark $F_r$.

**Liar!**

The success-rates for SPDecimate look different.
Question

Can we derive an MP algorithm that enables MPDecimate to solve all \( F \) with \( r \in [3.80, 4.20] \)?
Question

*Can we derive an MP algorithm that enables MPDecimate to solve all $F$ with $r \in [3.80, 4.20]$?*

- an idea from the literature brought forward by E. Maneva, E. Aurell, R. Zecchina, A. Braunstein and *many others*: interpolate BP and SP
**SAT Solving with Message Passing**

**Question**

*Can we derive an MP algorithm that enables $MPDecimate$ to solve all $F$ with $r \in [3.80, 4.20]$?*

- idea from the literature: interpolate BP and SP
  - $\rhoSP$ with interpolation parameter $\rho \in [0.0, 1.0]$
  - $\rhoSP$ can **simulate** SP \[ \forall v \in \text{Var}(F) : \beta_{\rhoSP}(v, \rho = 1) = \beta_{SP}(v) \]
Question

Can we derive an MP algorithm that enables MPDecimate to solve all $F$ with $r \in [3.80, 4.20]$?

- idea from the literature: interpolate BP and SP
  - $\rho_{SP}$ with interpolation parameter $\rho \in [0.0, 1.0]$
  - $\rho_{SP}$ can simulate SP \[ \forall v \in \text{Var}(F) : \beta_{\rho_{SP}}(v, \rho = 1) = \beta_{SP}(v) \]
  - $\rho_{SP}$ cannot simulate BP \[ \forall v \in \text{Var}(F) : \beta_{\rho_{SP}}(v, \rho = 0) \neq \beta_{BP}(v) \]
Results

...of my own work
Results

Result 1

- devise a **notational frame** to represent MP algorithms
  - BP, SP, EMBPG, EMSPG, ...
  - based on 14 equations to analytically formalize MP algorithms
  - different MP algorithms differ in exactly 4 equations $\delta, \mu^+, \mu^-, \mu^*$
Results

Result 1

- devise a **notational frame** to represent MP algorithms
  - BP, SP, EMBPG, EMSPG, ...
  - based on 14 equations to analytically formalize MP algorithms
  - different MP algorithms differ in exactly 4 equations $\delta, \mu^+, \mu^-, \mu^*$
  - an interpolation between BP and SP follows with $i \in [0.0, 1.0]$
    
    $$(1 - i)\delta_{BP} + i\delta_{SP} = \delta_{iSP}$$
    $$(1 - i)\mu^+_{BP} + i\mu^+_{SP} = \mu^+_{iSP}$$
    $$(1 - i)\mu^-_{BP} + i\mu^-_{SP} = \mu^-_{iSP}$$
    $$(1 - i)\mu^*_{BP} + i\mu^*_{SP} = \mu^*_{iSP}$$

- $iSP$ can simulate BP with $i = 0$ as well as SP $i = 1$
Results

Result 1

- devise a **notational frame** to represent MP algorithms
  - BP, SP, EMBPG, EMSPG, ...
  - based on 14 equations to analytically formalize MP algorithms
  - different MP algorithms differ in exactly 4 equations $\delta, \mu^+, \mu^-, \mu^*$
  - an interpolation between BP and SP follows with $i \in [0.0, 1.0]$

\[
\begin{align*}
(1 - i)\delta_{BP} + i\delta_{SP} &= \delta_{iSP} \\
(1 - i)\mu^+_{BP} + i\mu^+_{SP} &= \mu^+_{iSP} \\
(1 - i)\mu^-_{BP} + i\mu^-_{SP} &= \mu^-_{iSP} \\
(1 - i)\mu^*_{BP} + i\mu^*_{SP} &= \mu^*_{iSP}
\end{align*}
\]

- iSP can simulate BP with $i = 0$ as well as SP $i = 1$
Results

Result 1

- devise a **notational frame** to represent MP algorithms
  - BP, SP, EMBPG, EMSPG, ...
  - based on 14 equations to analytically formalize MP algorithms
  - different MP algorithms differ in exactly 4 equations $\delta, \mu^+, \mu^-, \mu^*$
  - an interpolation between BP and SP follows with $i \in [0.0, 1.0]$
    \begin{align*}
      (1 - i)\delta_{BP} + i\delta_{SP} &= \delta_{iSP} \\
      (1 - i)\mu^+_{BP} + i\mu^+_{SP} &= \mu^+_{iSP} \\
      (1 - i)\mu^-_{BP} + i\mu^-_{SP} &= \mu^-_{iSP} \\
      (1 - i)\mu^*_{BP} + i\mu^*_{SP} &= \mu^*_{iSP}
    \end{align*}
  - $i$SP can simulate BP with $i = 0$ as well as SP $i = 1$
Results

Result 1

- interpolations form a hierarchy
Results

Result 1

- interpolations form a hierarchy
Results

Result 1

- interpolations form a hierarchy

```
  iSP       iEMSPG
  /       \       /
 BP     SP    EMBPG   EMSPG
```
**Result 1**

- Interpolations form a **hierarchy**

1. The hierarchy shown here is incomplete.
2. IjMP can simulate all other MP algorithms shown here, which can now be understood as special cases of IjMP.
Results

Result 2

- extend the **notational frame** due to ideas from physics
  - others determined that BP and SP can be derived with approximation methods used in statistical mechanics
  - approximations rely on a variety of physical parameters such as
    - the temperature parameter $\tau$
    - the external magnetic field parameter $\eta$
    - the coupling-strength parameter $\psi$
    - the cavity-field parameter $\kappa$
  - translate these into the notational frame
    - by altering three of the 14 equations, including $\omega$

\[
\omega^t_{\text{MP}}(c, l) = \prod_{l' \in c \setminus \{l\}} \delta^t_{\text{MP}}(l', c)
\]
Results

Result 2

- extend the **notational frame** due to ideas from physics
  - BP and SP can be derived with approximation methods used in statistical mechanics
  - approximations rely on a variety of physical parameters such as
    - the temperature parameter $\tau$
    - the external magnetic field parameter $\eta$
    - the coupling-strength parameter $\psi$
    - the cavity-field parameter $\kappa$
  - translate these into the notational frame
    - by altering three of the 14 equations, including $\omega$

$$\omega^t_{\text{MP}}(c, l, (\tau, \eta, \psi, \kappa)) = \ldots$$
Results

Result 2

- extend the notational frame due to ideas from physics
  - BP and SP can be derived with approximation methods used in statistical mechanics
  - approximations rely on a variety of physical parameters such as
    - the temperature parameter $\tau$
    - the external magnetic field parameter $\eta$
    - the coupling-strength parameter $\psi$
    - the cavity-field parameter $\kappa$
  - translate these into the notational frame
    - by altering three of the 14 equations, including $\omega$

$$\omega^t_{MP}(c, l, (\tau, \eta, \psi, \kappa)) = (1 - \tau) \left\{ (1 - |\eta|) \prod_{l' \in c \setminus \{l\}} \left[ \delta^t_{MP}(l', c, (\tau, \eta, \psi, \kappa)) \right] \frac{1}{\psi_{l'}} + \frac{|\eta| + \text{sgn}(l) \cdot \eta}{2} \right\}$$
Results

Result 2

- extensions lead to the *extended hierarchy*
Results

Result 2

- extensions lead to the extended hierarchy
Result 2

- extensions lead to the **extended hierarchy**

- ijMPP can simulate all other MP algorithms shown here
Results

Result 2

- \textit{ijMPP} leads to the \textit{ijMPPDecide}(\(\alpha, F\)) heuristic
- that can simulate other heuristics (provide the same decisions)
- while assuming an appropriate setting to \(i, j, \tau, \eta, \psi, \kappa\)
Results

Result 2

- $ijMPP$ leads to the $ijMPP\text{Decide}(\alpha, F)$ heuristic
- that can simulate other heuristics (provide the same decisions)
- while assuming an appropriate setting to $i, j, \tau, \eta, \psi, \kappa$
  - random decision making $\text{Rand}$
  - random-variable 1-assignment first $\text{RandOneFirst}$
  - random-variable 0-assignment first $\text{RandZeroFirst}$
  - negative-imbalance 0-first $\text{NIZeroFirst}$
  - positive-imbalance 1-first $\text{PIOneFirst}$
  - overall-imbalance conflict-avoiding $\text{OICA}$
  - overall-imbalance conflict-seeking $\text{OICS}$
  - overall-score conflict-avoiding $\text{OSCA}$

- none of these heuristics have anything to do with MP algorithms
- now they are all understood as special cases of $ijMPP\text{Decide}(\alpha, F)$
Results

Result 3

- $ijMPP\text{Decide}(\alpha, F)$ provides opportunities for parameter tuning
  - statistically significant performance improvements possible
  - despised by some colleagues (too much try-and-error)
Results

Result 3

- \( ij\text{MPPDecide}(\alpha, F) \) provides opportunities for parameter tuning
  - statistically significant performance improvements possible
  - despised by some colleagues (too much try-and-error)
- winning medals and certificates in the SAT Competition
Summary and Conclusions

- the \( k \)-SAT problem, SAT solving, heuristics for SAT solving
- Message Passing algorithms
- interpolate MP algorithms into more flexible versions
  - that enable SAT solvers to solve a wider range of formulas
- major results
  1. provide an approach to interpolate MP algorithms
     - interpolation-based MP algorithms can simulate what they interpolate
     - which lead to a hierarchy of MP algorithms
  2. extend MP algorithms by introducing physical parameters
     - physically extended versions can simulate their original counterparts
     - which lead to an extended MP hierarchy
     - and allowed us to obtain the flexible ijMPP algorithm
     - the resulting ijMPPDecide heuristic allowed us to uncover theoretical connections between non-MP heuristics
  3. ijMPPDecide heuristic
     - yield possibilities for parameter tuning and enables statistically significant performance improvements in practice
     - medals
Publications

- Gableske, O.: The Effect of Clause Elimination on SLS for SAT, POS 2012.

George Boole and Ludwig E. Boltzmann

Thank you very much!

Non-rigorous arguments suggest that there will be questions.