

A Simple $\frac{3}{4}$ -Approximation Algorithm for MAX SAT

David P. Williamson



Joint work with Matthias Poloczek (Cornell), Georg Schnitger
(Frankfurt), and Anke van Zuylen (William & Mary)

Maximum Satisfiability

- Input:

n Boolean variables x_1, \dots, x_n

m clauses C_1, \dots, C_m with weights $w_j \geq 0$

– each clause is a disjunction of literals,

e.g. $C_1 = x_1 \vee x_2 \vee \bar{x}_3$

- Goal: truth assignment to the variables that maximizes the weight of the satisfied clauses

Approximation Algorithms

- An α -approximation algorithm runs in polynomial time and returns a solution of at least α times the optimal.
- For a randomized algorithm, we ask that the expected value is at least α times the optimal.

A $\frac{1}{2}$ -approximation algorithm

- Set each x_i to true with probability $\frac{1}{2}$.
- Then if l_j is the number of literals in clause j

$$E[\text{Weight satisfied clauses}]$$

$$= \sum_{j=1}^m w_j \Pr[\text{Clause } j \text{ satisfied}]$$

$$= \sum_{j=1}^m w_j \left(1 - \left(\frac{1}{2} \right)^{l_j} \right)$$

$$\geq \frac{1}{2} \sum_{j=1}^m w_j \geq \frac{1}{2} OPT.$$

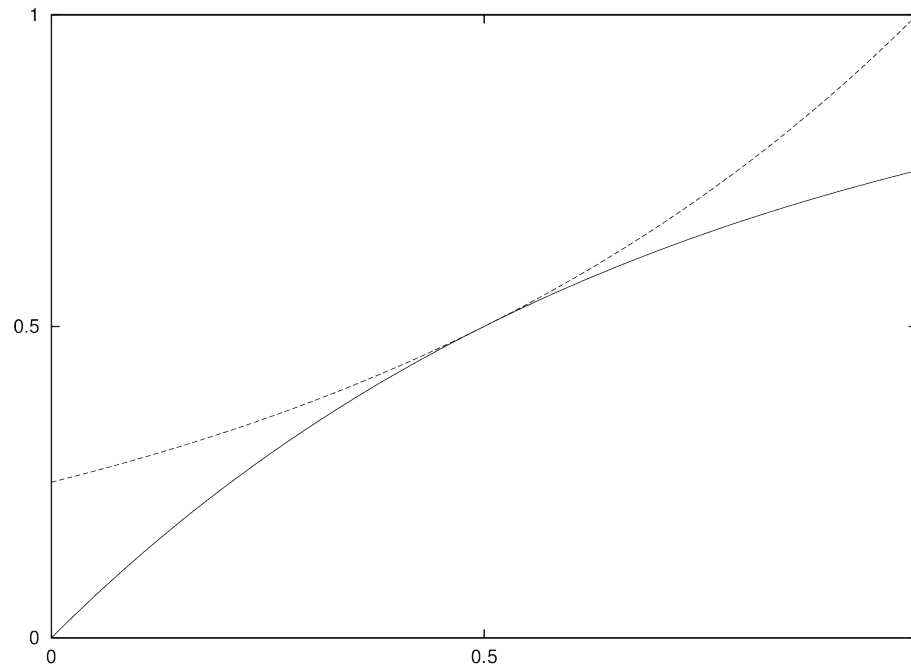
What about a deterministic algorithm?

- Use the *method of conditional expectations* (Erdős and Selfridge '73, Spencer '87)
- If $E[W|x_1 \leftarrow true] \geq E[W|x_1 \leftarrow false]$ then set x_1 true, otherwise false.
- Similarly, if X_{i-1} is event of how first $i - 1$ variables are set, then if $E[W|X_{i-1}, x_i \leftarrow true] \geq E[W|X_{i-1}, x_i \leftarrow false]$, set x_i true.
- Show inductively that $E[W|X_i] \geq E[W] \geq \frac{1}{2} \text{OPT}$.

An LP relaxation

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^m w_j z_j \\ &\text{subject to} && \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, && \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i, \\ & && 0 \leq y_i \leq 1, && i = 1, \dots, n, \\ & && 0 \leq z_j \leq 1, && j = 1, \dots, m. \end{aligned}$$

Nonlinear randomized rounding



(Goemans, W 94) Pick any function f such that $1 - 4^{-x} \leq f(x) \leq 4^{x-1}$. Set x_i true with probability $f(y_i^*)$, where y^* is an optimal LP solution.

Analysis

$$\begin{aligned}\Pr[\text{clause } C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - f(y_i^*)) \prod_{i \in N_j} f(y_i^*) \\ &\leq \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^* - 1} \\ &= 4^{-\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*)\right)} \\ &\leq 4^{-z_j^*}.\end{aligned}$$

$$\begin{aligned}E[W] &\geq \sum_{j=1}^m w_j \Pr[\text{clause } C_j \text{ satisfied}] \\ &\geq \sum_{j=1}^m w_j \left(1 - 4^{-z_j^*}\right) \\ &\geq \frac{3}{4} \sum_{j=1}^m w_j z_j^* \geq \frac{3}{4} OPT.\end{aligned}$$

Integrality gap

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^m w_j z_j \\ &\text{subject to} && \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, && \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i, \\ &&& 0 \leq y_i \leq 1, && i = 1, \dots, n, \\ &&& 0 \leq z_j \leq 1, && j = 1, \dots, m. \end{aligned}$$

$$x_1 \vee x_2, \quad \bar{x}_1 \vee x_2, \quad x_1 \vee \bar{x}_2, \quad \bar{x}_1 \vee \bar{x}_2$$

The result is tight since LP solution $z_1 = z_2 = z_3 = z_4 = 1$ and $y_1 = y_2 = \frac{1}{2}$ feasible for instance above, but $\text{OPT} = 3$.

Chan, Lee, Raghavendra, Steurer (STOC 13) show no superpolynomially sized LP can give a better integrality gap.

Current status

- NP-hard to approximate better than 0.875 (Håstad '01)
- Combinatorial approximation algorithms
 - Johnson's algorithm (1974): Simple $\frac{1}{2}$ -approximation algorithm (Greedy version of the randomized algorithm)
 - Improved analysis of Johnson's algorithm: $\frac{2}{3}$ -approx. guarantee [Chen, Friesen, Zheng '99, Engebretsen '04]
 - Randomizing variable order improves guarantee slightly [Costello, Shapira, Tetali SODA 11]
- Algorithms using Linear or Semidefinite Programming
 - Yannakakis '94, Goemans, W '94:

Question [W '98]: **Is it possible to obtain a $\frac{3}{4}$ -approximation algorithm without solving a linear program?**

(Selected) results

- Poloczek, Schnitger (SODA 11):
 - “randomized Johnson” – combinatorial $\frac{3}{4}$ -approximation algorithm
- Van Zuylen (WAOA 11):
 - Simplification of “randomized Johnson” probabilities and analysis
- Buchbinder, Feldman, Naor, and Schwartz (FOCS 12):
 - Another $\frac{3}{4}$ -approximation algorithm for MAX SAT as a special case of submodular function maximization
 - Can be shown that their MAX SAT alg is equivalent to van Zuylen’s.

(Selected) results

- Poloczek, Schnitger '11
- Van Zuylen '11
- Buchbinder, Feldman, Naor and Schwartz '12

Common properties:

- iteratively set the variables in an “online” fashion,
- the probability of setting x_i to true depends on clauses containing x_i or \bar{x}_i that are still undecided.

Today

- Give “textbook” version of Buchbinder et al.’s algorithm with an even simpler analysis (Poloczek, van Zuylen, W, LATIN 14)
- Give a simple deterministic version of the algorithm (Poloczek, Schnitger, van Zuylen, W, manuscript)
- Give an experimental analysis that shows that the algorithm works very well in practice (Poloczek, W, SEA 2016)

Buchbinder et al.'s approach

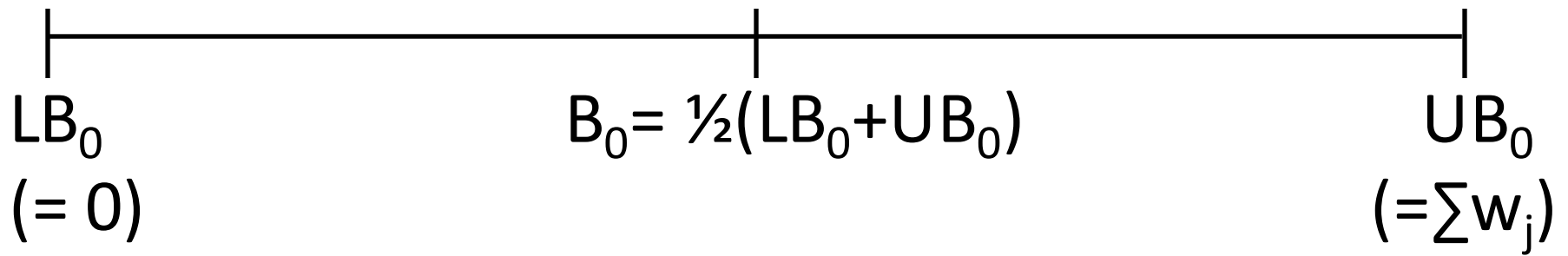
- Keep two bounds on the solution
 - **Lower bound LB** = weight of clauses already satisfied
 - **Upper bound UB** = weight of clauses not yet unsatisfied
- Greedy can focus on two things:
 - maximize **LB**,
 - maximize **UB**,

but either choice has bad examples...

$$\text{E.g. } x_1 \vee x_2 \text{ (wt } 1+\epsilon), \bar{x}_1 \text{ (wt } 1)$$

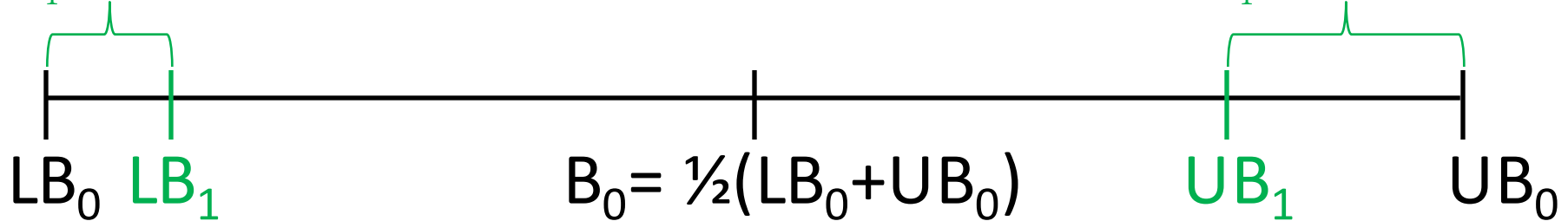
$$x_1 \vee x_2 \text{ (wt } 1+\epsilon), \bar{x}_1 \text{ (wt } \epsilon), \bar{x}_2 \text{ (wt } 1)$$

- Key idea: make choices to increase $\mathbf{B} = \frac{1}{2} (\mathbf{LB} + \mathbf{UB})$



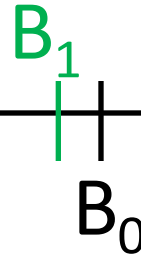
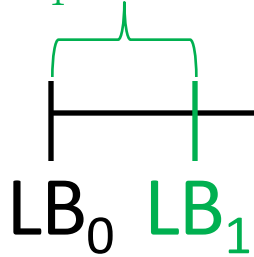
Weight of
undecided
clauses
satisfied by
 $x_1 = \text{true}$

Weight of
undecided
clauses
unsatisfied by
 $x_1 = \text{true}$

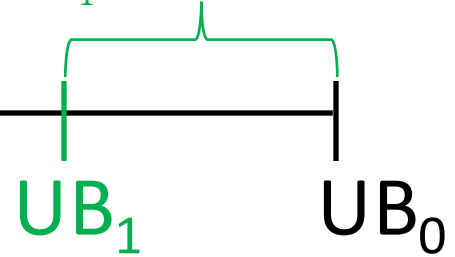


Set x_1 to true

Weight of
undecided
clauses
satisfied by
 $x_1 = \text{true}$



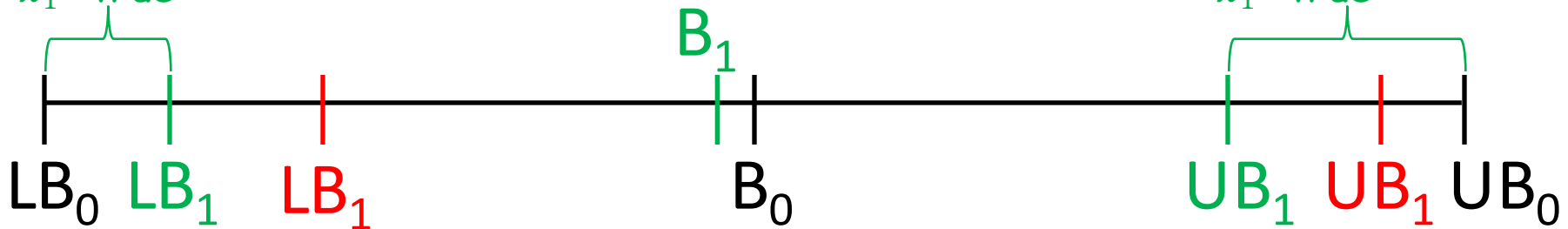
Weight of
undecided
clauses
unsatisfied by
 $x_1 = \text{true}$



Set x_1 to true

Weight of undecided clauses satisfied by $x_1 = \text{true}$

Weight of undecided clauses unsatisfied by $x_1 = \text{true}$



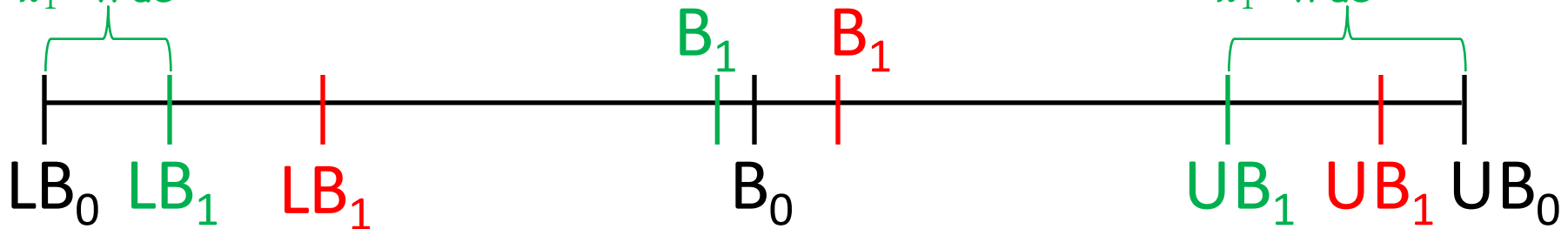
Set x_1 to true

or

Set x_1 to false

Weight of undecided clauses satisfied by $x_1 = \text{true}$

Weight of undecided clauses unsatisfied by $x_1 = \text{true}$



Set x_1 to true

or

Set x_1 to false

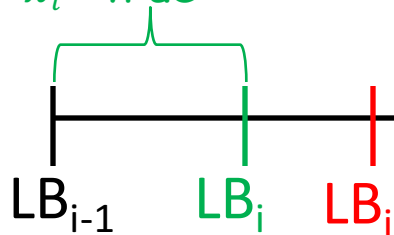
Guaranteed that

$$(B_1 - B_0) + (B_1 - B_0) \geq 0$$

t_1

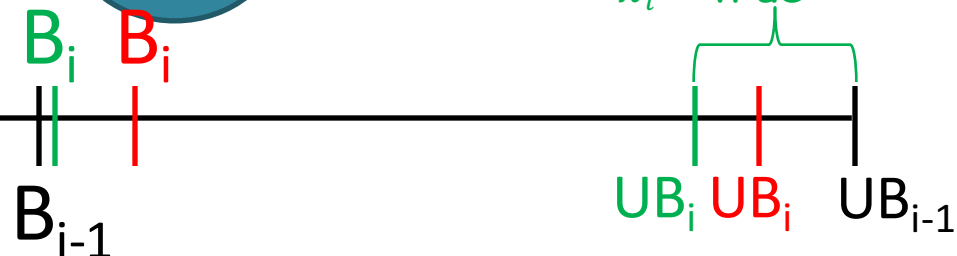
f_1

Weight of undecided clauses satisfied by $x_i = \text{true}$



Remark: This is the algorithm proposed independently by BFNS'12 and vZ'11

Weight of undecided clauses unsatisfied by $x_i = \text{true}$



Algorithm:

- if $t_i < 0$, set x_i to **false**
- if $f_i < 0$, set x_i to **true**
- else, set x_i to true with probability $\frac{t_i}{t_i + f_i}$

$$\underbrace{(B_i - B_{i-1})}_{t_i} + \underbrace{(B_i - B_{i-1})}_{f_i} \geq 0$$

Example

Initialize:

- $LB = 0$
- $UB = 6$

Step 1:

- $t_1 = \frac{1}{2} (\Delta LB + \Delta UB) = \frac{1}{2} (1 + (-2)) = -\frac{1}{2}$
- $f_1 = \frac{1}{2} (\Delta LB + \Delta UB) = \frac{1}{2} (2 + 0) = 1$
- Set x_1 to false

Clause	Weight
\bar{x}_1	2
$x_1 \vee x_2$	1
$\bar{x}_2 \vee x_3$	3

Example

Clause	Weight
\bar{x}_1	2
x_1 $\vee x_2$	1
$\bar{x}_2 \vee x_3$	3

Step 2:

- $t_2 = \frac{1}{2} (\Delta LB + \Delta UB) = \frac{1}{2} (1 + 0) = \frac{1}{2}$
- $f_2 = \frac{1}{2} (\Delta LB + \Delta UB) = \frac{1}{2} (3 + (-1)) = 1$
- Set x_2 to true with probability $1/3$ and to false with probability $2/3$

Example

Clause	Weight
\bar{x}_1	2
$x_1 \vee x_2$	1
$\bar{x}_2 \vee x_3$	3

Algorithm's solution:

$$x_1 = \text{false}$$

$$x_2 = \text{true w.p. } 1/3 \text{ and false w.p. } 2/3$$

$$x_3 = \text{true}$$

Expected weight of satisfied clauses: $5\frac{1}{3}$

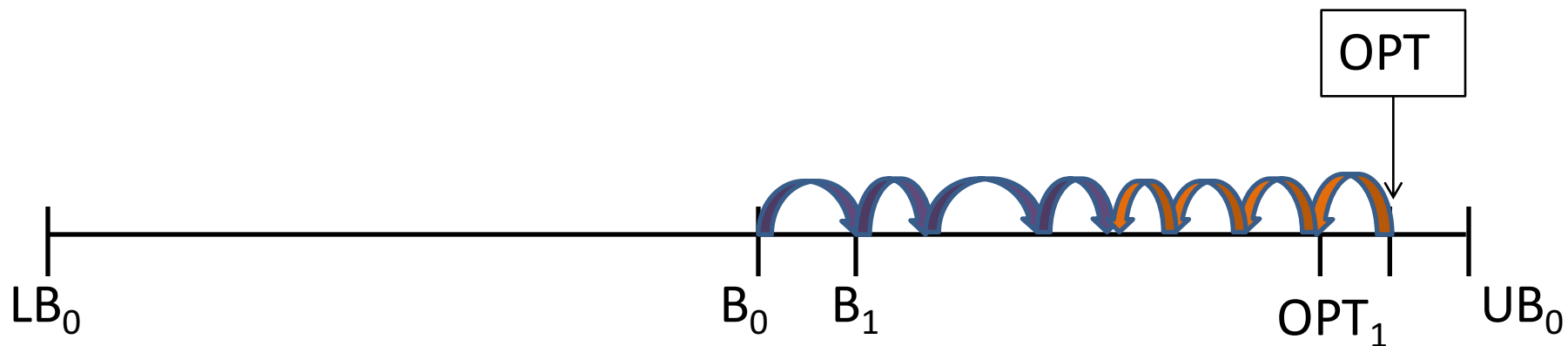
Relating Algorithm to Optimum

Let $x_1^*, x_2^*, \dots, x_n^*$ be an optimal truth assignment

Let $OPT_i =$ weight of clauses satisfied if setting x_1, \dots, x_i as the algorithm does, and $x_{i+1} = x_{i+1}^*, \dots, x_n = x_n^*$

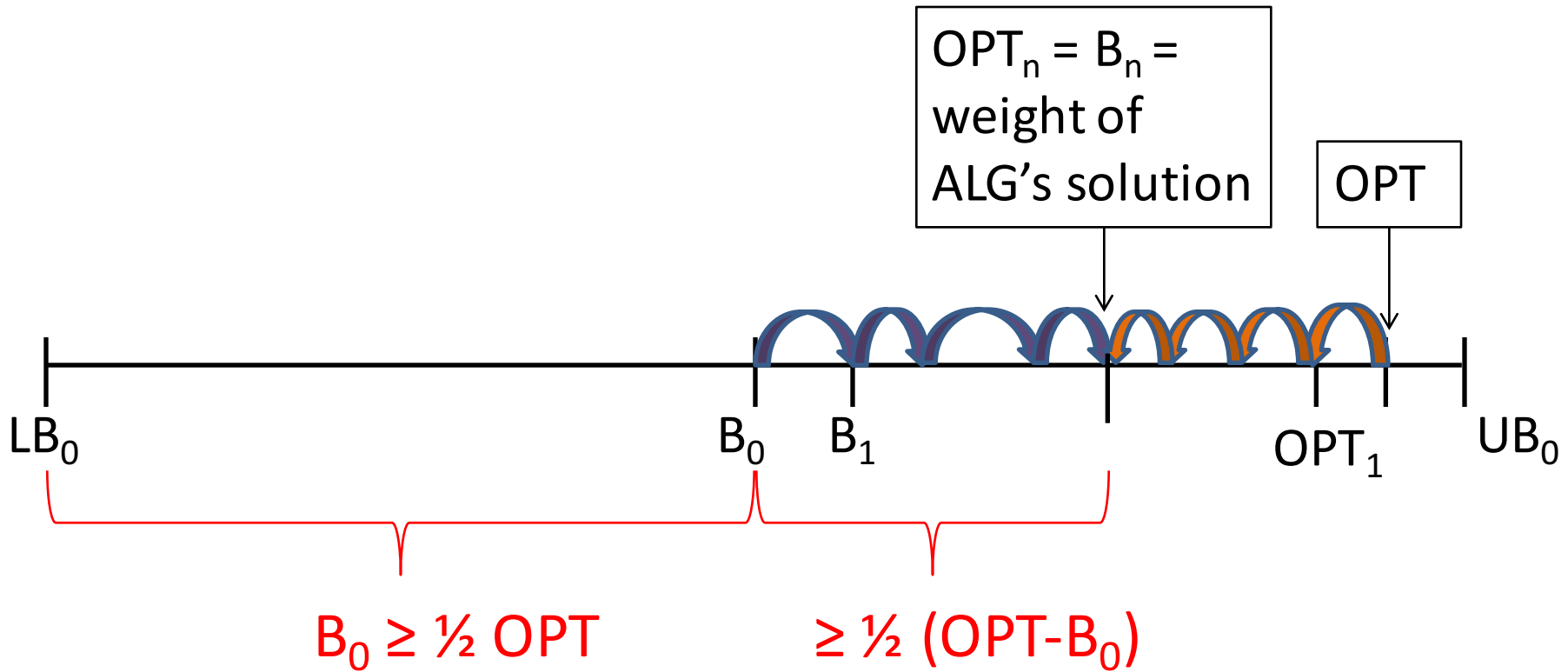
Key Lemma:

$$E[B_i - B_{i-1}] \geq E[OPT_{i-1} - OPT_i]$$



Key Lemma:

$$E[B_i - B_{i-1}] \geq E[OPT_{i-1} - OPT_i]$$

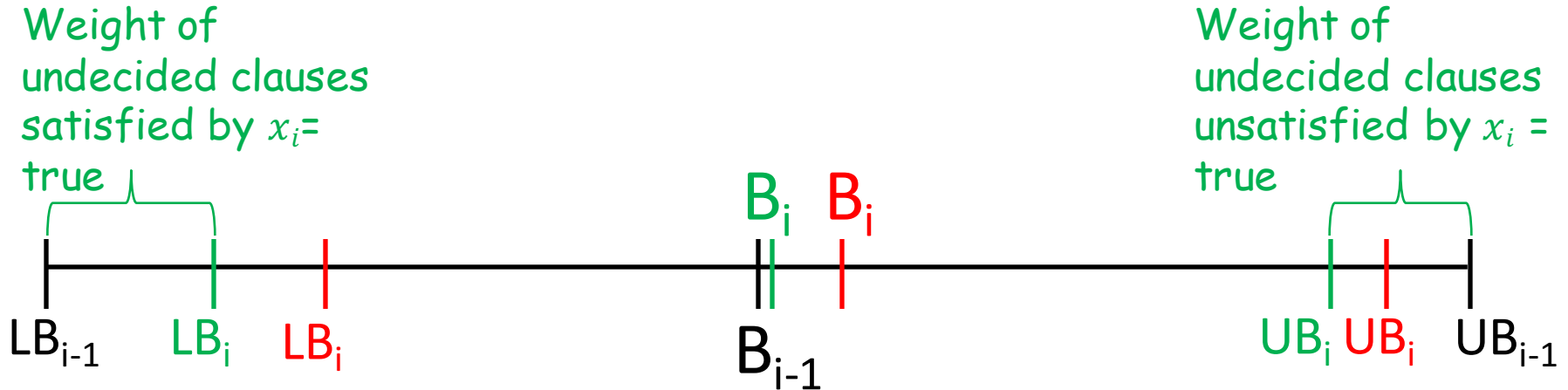


Key Lemma:

Conclusion: expected weight of ALG's solution is

$$E[B_n] \geq B_0 + \frac{1}{2} (OPT - B_0) = \frac{1}{2} (OPT + B_0) \geq \frac{3}{4} OPT$$

Relating Algorithm to Optimum



Suppose $x_i^* = \text{true}$

If algorithm sets x_i to **true**,

- $B_i - B_{i-1} = t_i$
- $OPT_{i-1} - OPT_i = 0$

If algorithm sets x_i to **false**,

- $B_i - B_{i-1} = f_i$
- $OPT_{i-1} - OPT_i \leq LB_i - LB_{i-1} + (UB_i - UB_{i-1})$
 $= 2(B_i - B_{i-1}) = 2t_i$

Want to show:

Key Lemma:

$$E[B_i - B_{i-1}] \geq E[OPT_{i-1} - OPT_i]$$

Relating Algorithm to Optimum

Want to show:

Key Lemma:

$$E[B_i - B_{i-1}] \geq E[OPT_{i-1} - OPT_i]$$

Know:

If algorithm sets x_i to **true**,

- $B_i - B_{i-1} = t_i$
- $OPT_{i-1} - OPT_i = 0$

If algorithm sets x_i to **false**,

- $B_i - B_{i-1} = f_i$
- $OPT_{i-1} - OPT_i \leq 2t_i$

Case 1: $f_i < 0$ (algorithm sets x_i to **true**):

$$E[B_i - B_{i-1}] = t_i > 0 = E[OPT_{i-1} - OPT_i]$$

Case 2: $t_i < 0$ (algorithm sets x_i to **false**):

$$E[B_i - B_{i-1}] = f_i > 0 > 2t_i \geq E[OPT_{i-1} - OPT_i]$$

Relating Algorithm to Optimum

Want to show:

Key Lemma:

$$E[B_i - B_{i-1}] \geq E[OPT_{i-1} - OPT_i]$$

Know:

If algorithm sets x_i to true,

- $B_i - B_{i-1} = t_i$
- $OPT_{i-1} - OPT_i = 0$

Equal to $(t_i - f_i)^2 + 2t_i f_i$ false,
 t_i

Case 3: $t_i \geq 0, f_i \geq 0$ (algorithm sets x_i to true w.p. $t_i / (t_i + f_i)$):

$$E[B_i - B_{i-1}] = t_i \frac{t_i}{t_i + f_i} + f_i \frac{f_i}{t_i + f_i} = \frac{1}{t_i + f_i} (t_i^2 + f_i^2)$$

$$E[OPT_{i-1} - OPT_i] \leq 0 \frac{t_i}{t_i + f_i} + 2t_i \frac{f_i}{t_i + f_i} = \frac{1}{t_i + f_i} (2t_i f_i)$$

Question

Is there a simple combinatorial deterministic $\frac{3}{4}$ -approximation algorithm?

Deterministic variant?

Greedy maximizing B_i is not good enough:

Clause	Weight
x_1	1
$\bar{x}_1 \vee x_2$	$2+\epsilon$
x_2	1
$\bar{x}_2 \vee x_3$	$2+\epsilon$
.....	
x_{n-1}	1
$\bar{x}_{n-1} \vee x_n$	$2+\epsilon$

Optimal assignment sets
all variables to true
 $OPT = (n-1)(3+\epsilon)$

Greedy increasing B_i
sets variables
 x_1, \dots, x_{n-1} to false
 $GREEDY = (n-1)(2+\epsilon)$

A negative result

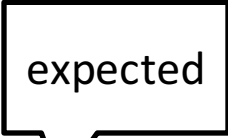
Poloczek (ESA 11): No deterministic “priority algorithm” can be a $\frac{3}{4}$ -approximation algorithm, using scheme introduced by Borodin, Nielsen, and Rackoff ‘03.

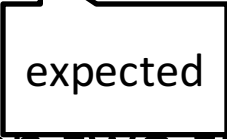
- Algorithm makes one pass over the variables and sets them.
- Only looks at weights of clauses in which current variable appears positively and negatively (not at the other variables in such clauses).
- Restricted in information used to choose next variable to set.

But...

- It is possible...
- ... with a two-pass algorithm (Thanks to Ola Svensson).
- First pass: Set variables x_i fractionally (i.e. probability that x_i true), so that $E[W] \geq \frac{3}{4} OPT$.
- Second pass: Use method of conditional expectations to get deterministic solution of value at least as much.

Buchbinder et al.'s approach

- Keep two bounds  fractional solution
 - **Lower bound LB** = weight of clauses already satisfied
 - **Upper bound UB** = weight of clauses not yet unsatisfied

- Greedy can focus on  two things:
 - maximize **LB**,
 - maximize **UB**,

but either choice has bad example 

- Key idea: make choices to increase **B** = $\frac{1}{2} (\mathbf{LB} + \mathbf{UB})$

As before

Let t_i be (expected) increase in bound B_{i-1} if we set x_i true; f_i be (expected) increase in bound if we set x_i false.

Algorithm:

For $i \leftarrow 1$ to n

- if $t_i < 0$, set x_i to 0
- if $f_i < 0$, set x_i to 1
- else, set x_i to $\frac{t_i}{t_i + f_i}$

For $i \leftarrow 1$ to n

- If $E[W | X_{i-1}, x_i \leftarrow true] \geq E[W | X_{i-1}, x_i \leftarrow false]$, set x_i true
- Else set x_i false

Analysis

- Proof that after the first pass $E[W] \geq \frac{3}{4} OPT$ is almost the same as before.
- Proof that final solution output has value at least $E[W] \geq \frac{3}{4} OPT$ is via method of conditional expectation.
- Algorithm can be implemented in linear time.

Experimental Analysis

- How well do these algorithms work on structured instances?
- How do they compare to other types of algorithms (e.g. local search)?
- Can we use the randomization to our advantage?

The Instances

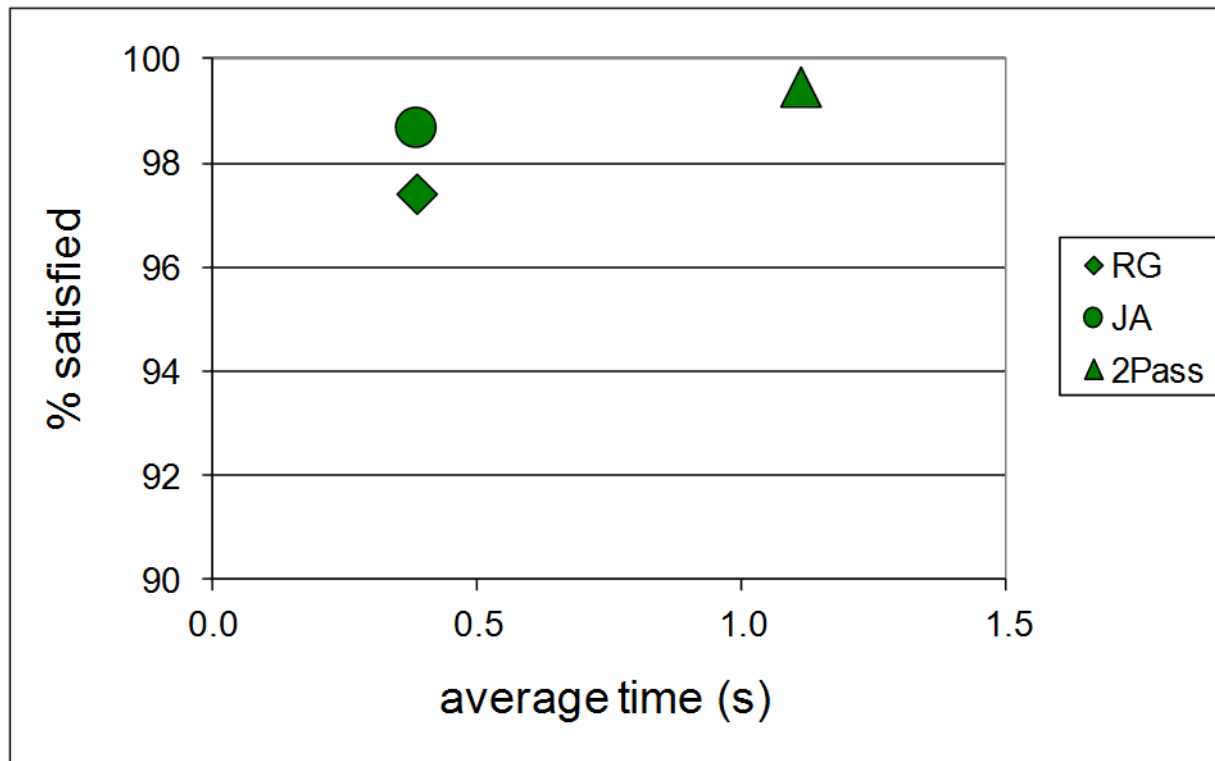
- From SAT and MAX SAT competitions in 2014 and 2015, all unweighted:
 - Industrial/applications: formal verification, crypto attacks, etc (300 + 55 instances)
 - Crafted: Max cut, graph isomorphism, etc (300 + 402 instances)
 - Random: With various ratios of clauses/variables (225 + 702 instances)
- Sizes:
 - Average for industrial: .5M variables in 2M clauses
 - Largest: 14M in 53M clauses
 - Larger in SAT instances than MAX SAT

The Measure

- Rather than approximation ratio, we use the *totality ratio*, ratio of satisfied clauses to the number of clauses in the input.

Greedy Algorithms

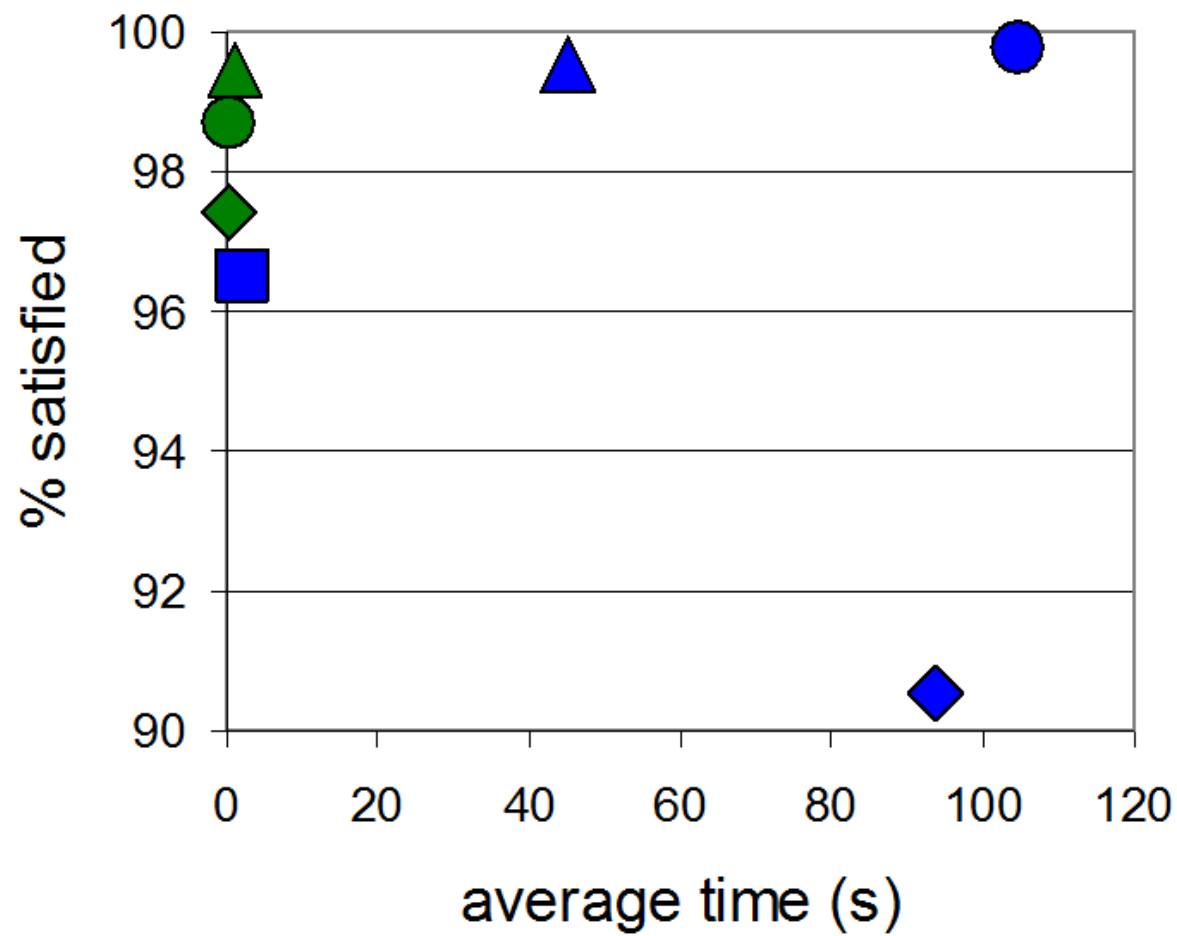
SAT/Industrial instances: Johnson's algorithm (JA) versus Randomized Greedy (RG) versus the 2-pass algorithm (2Pass).



Local Search

We compared the greedy algorithms versus a number of local search algorithms studied by Pankratov and Borodin (SAT 2010).

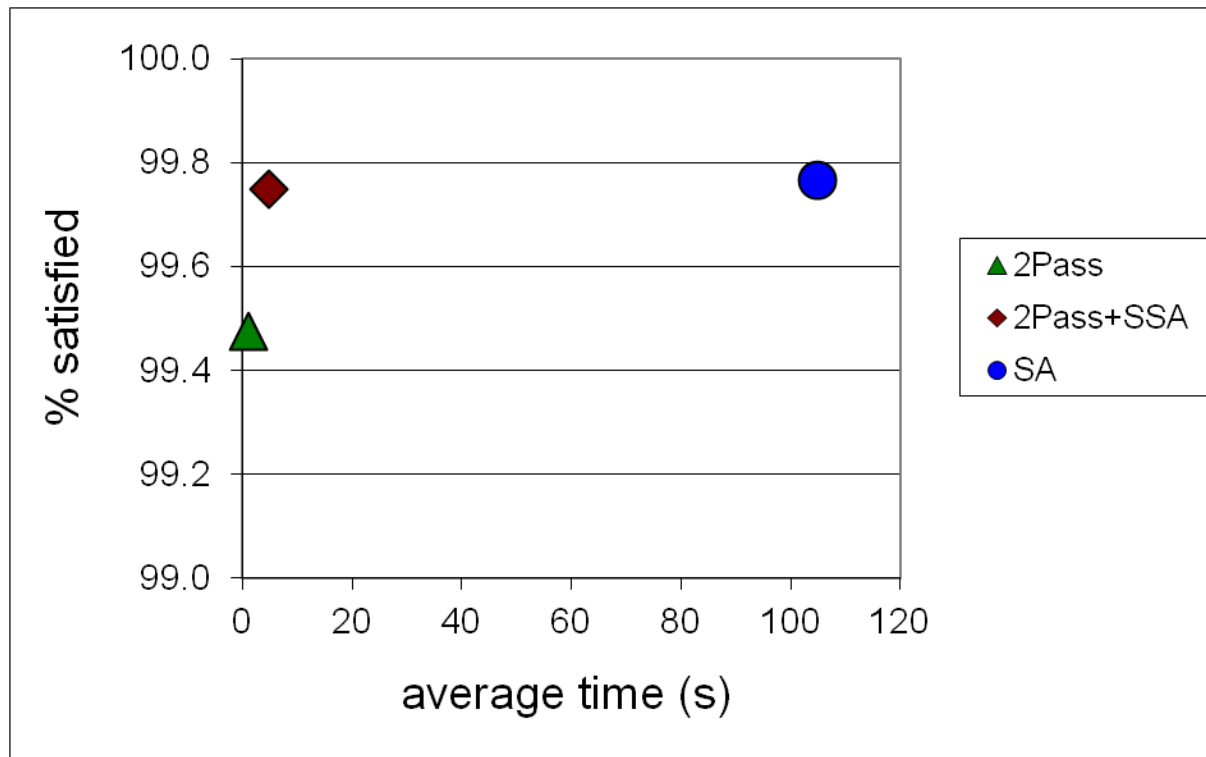
- WalkSAT: Selman, Kautz, Cohen (1993), Kautz (2014)
- Non-Oblivious Local Search (NOLS): Khanna, Motwani, Sudan, Vazirani (1998)
- Simulated Annealing (SA): Spears (1993)



- ◆ RG
- JA
- ▲ 2Pass
- WalkSat
- ▲ 2Pass+NOLS
- ◆ NOLS+TS
- SA

A Hybrid Algorithm

Adding the last 10 iterations of simulated annealing on top of 2-Pass worked really well, not that much slower. The last 10 iterations by themselves was slightly faster, only slightly worse.



Randomization

- Suppose we randomize over the variable orderings? Costello, Shapira, and Tetali (SODA 11) show this improves the worst-case performance of Johnson's algorithm.
- For industrial instances, this makes the performance of the greedy algorithms worse: Johnson's alg from 98% to 95.8%, RG from 95.7% to 92.8%.

Randomization

- What about multiple trials of RG (10x)?
- Increases average fraction of satisfied clause by only 0.07%.

Conclusion

- We show this two-pass idea works for other problems as well (e.g. deterministic $\frac{1}{2}$ -approximation algorithm for MAX DICUT, MAX NAE SAT).
- Can we characterize the problems for which it does work?

Conclusion

- More broadly, are there other places in which we can reduce the computation needed for approximation algorithms and make them practical?
 - E.g. Trevisan 13/Soto 15 give a .614-approximation algorithm for Max Cut using a spectral algorithm.
 - Can we beat $\frac{3}{4}$ using a spectral algorithm?
 - For just MAX 2SAT?
 - We can get .817 for *balanced* instances (Paul, Poloczek, W LATIN 16)
 - Curiously, the algorithm seems to beat the GW SDP algorithm on average in practice (Paul et al.)

Thanks for your time and attention.