A Simple, Greedy Approximation Algorithm for MAX SAT

David P. Williamson

Joint work with Matthias Poloczek (Frankfurt, Cornell)

and Anke van Zuylen (William & Mary)

Greedy algorithms

"Greed, for lack of a better word, is good. Greed is right. Greed works." – Gordon Gekko, *Wall Street*

"Greedy algorithms work." – Alan Hoffman, *IBM*

Another reason

- When I interviewed at Watson, half of my talk was about maximum satisfiability, the other half about the max cut SDP result.
- I thought, “Oh no, I have to talk about
  - Hardness of approximation in front of Madhu Sudan,
  - Randomized rounding in front of Prabhakar Raghavan,
  - And eigenvalue bounds in front of Alan Hoffman.”
- Today I revisit the first part of that talk.

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 ½-approximation algorithm

- Set each x_i to true with probability ½.
- Then if l_j is the number of literals in clause j

$$
\begin{aligned}
& E[\text{Weight satisfied clauses}] \\
& \quad = \quad \sum_{j=1}^{m} w_j \Pr[\text{Clause } j \text{ satisfied}] \\
& \quad = \quad \sum_{j=1}^{m} w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\
& \quad \geq \quad \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT.
\end{aligned}
$$

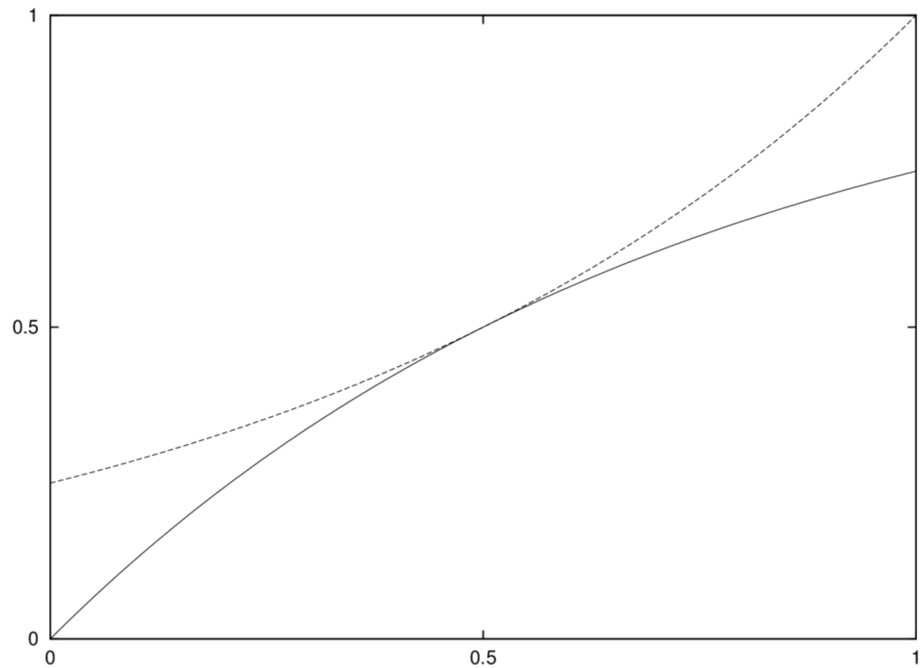
What about a deterministic algorithm?

- 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 $E[W|X_{i-1}, x_i \leftarrow true] \geq E\left[W \middle| X_{i-1,} x_i \leftarrow false\right]$, set x_i true.
- Show inductively that $E[W|X_i] \geq E[W] \geq \frac{1}{2}$ OPT.

An LP relaxation

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

Randomized rounding



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

Analysis

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

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

Integrality gap

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

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

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 OPT = 3.

Current status

- NP-hard to approximate better than 0.875 (Håstad '01)

- Combinatorial approximation algorithms
  - Johnson's algorithm (1974): Simple ½-approximation algorithm (Greedy version of the randomized algorithm)
  - Improved analysis of Johnson's algorithm: $^2/_3$-approx. guarantee [Chen-Friesen-Zheng '99, Engebretsen '04]
  - Randomizing variable order improves guarantee slightly [Costello-Shapira-Tetali '11]

- Algorithms using Linear or Semidefinite Programming
  - Yannakakis '94, Goemans-W '94:

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

(Selected) recent results

- Poloczek-Schnitger ’11:
  - “randomized Johnson” – combinatorial ¾-approximation algorithm
- Van Zuylen ’11:
  - Simplification of “randomized Johnson” probabilities and analysis
  - Derandomization using Linear Programming
- Buchbinder, Feldman, Naor, and Schwartz ’12:
  - Another ¾-approximation algorithm for MAX SAT as a special case of submodular function maximization
  - We show MAX SAT alg is equivalent to van Zuylen ‘11.

(Selected) recent 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.

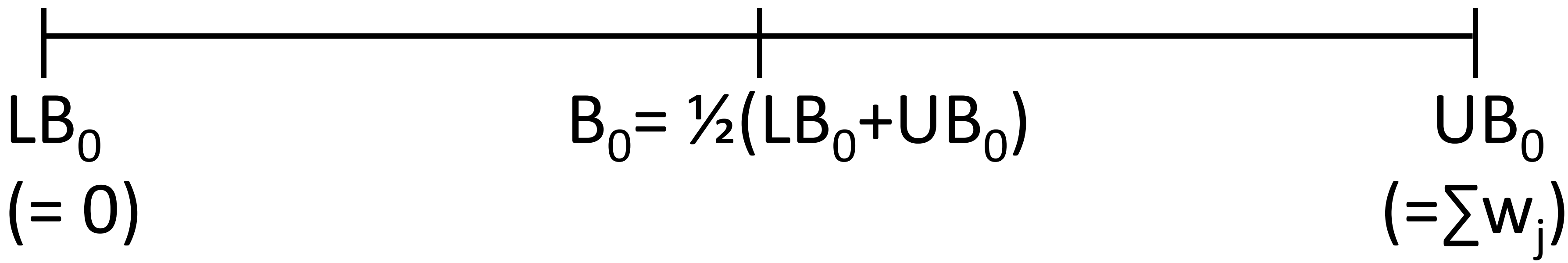
Today

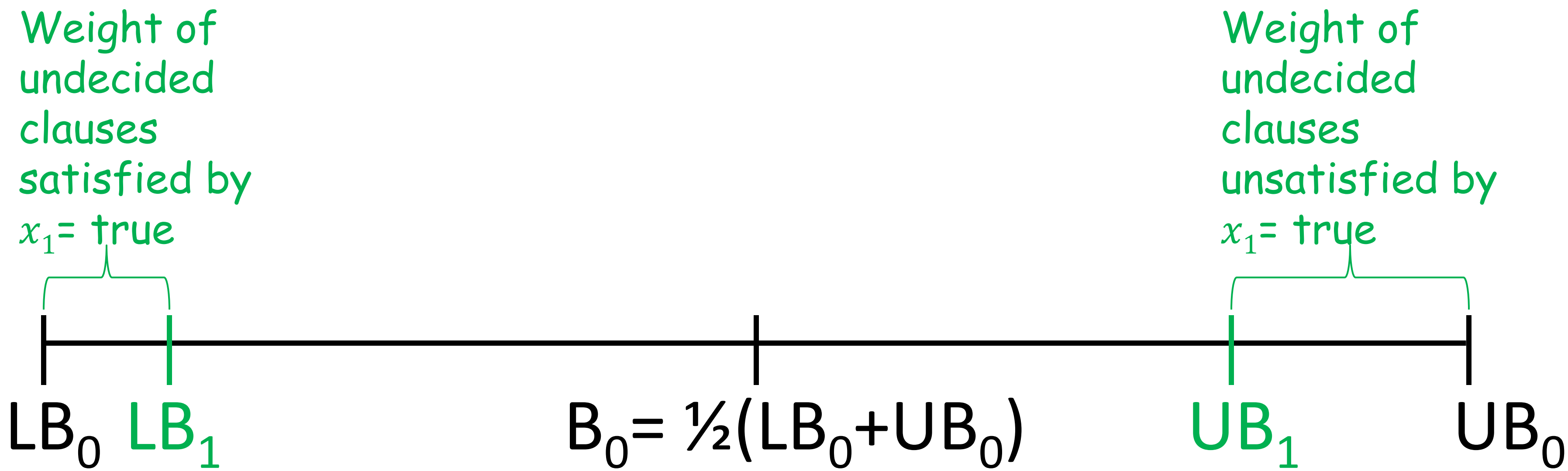
- Give “textbook” version of Buchbinder et al.’s algorithm with an even simpler analysis

Buchbinder et al.'s approach

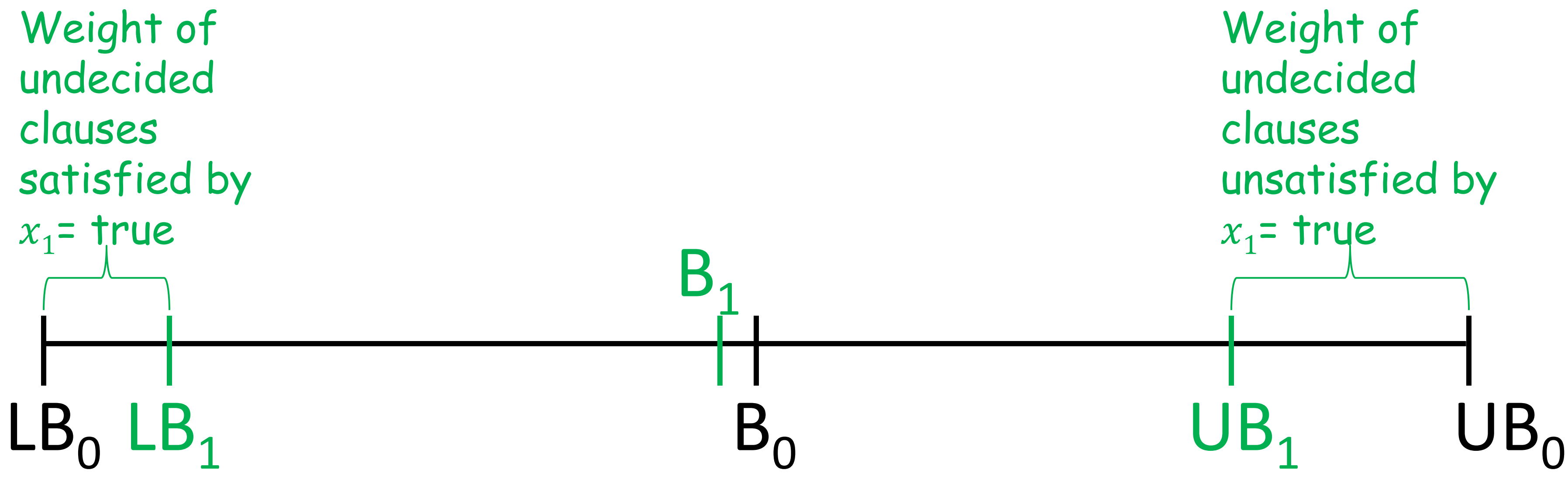
- Keep two bounds on the solution
  - **Lower bound LB** = weight of clauses already satisfied
  - **Upper bound UB** = weight of clauses not yet unsatisfied
- Greedy can focus on two things:
  - maximize **LB**,
  - maximize **UB**,

  but either choice has bad examples…
- Key idea: make choices to increase **B = ½ (LB+UB)**

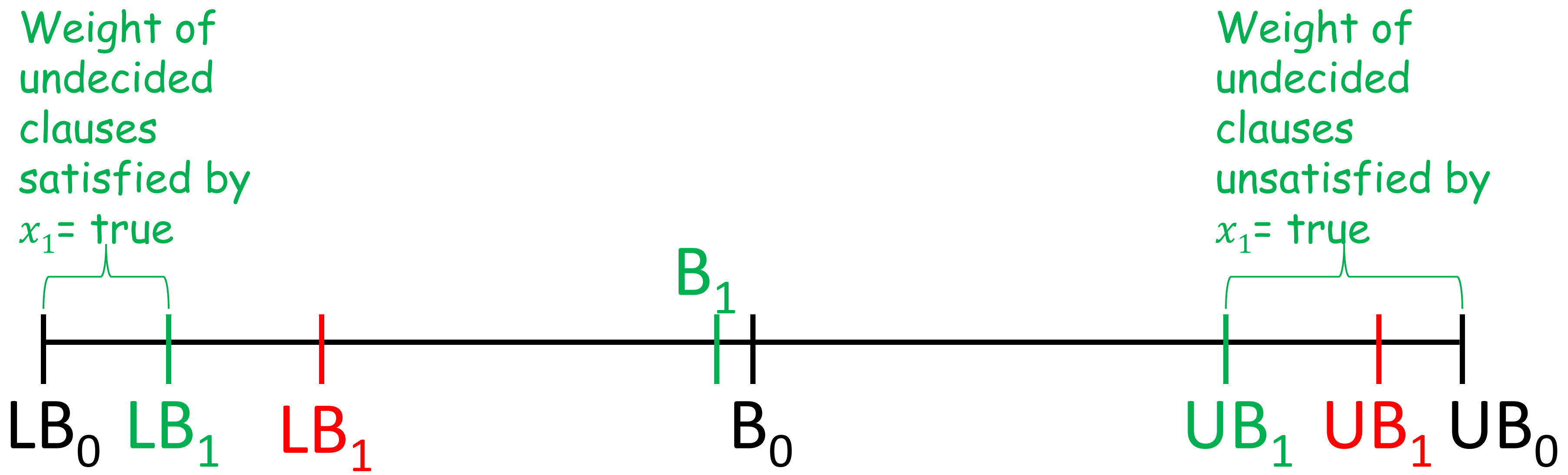

LB_0
$(= 0)$
$B_0 = ½(LB_0+UB_0)$
UB_0
$(=\sum w_j)$



Set x_1 to true



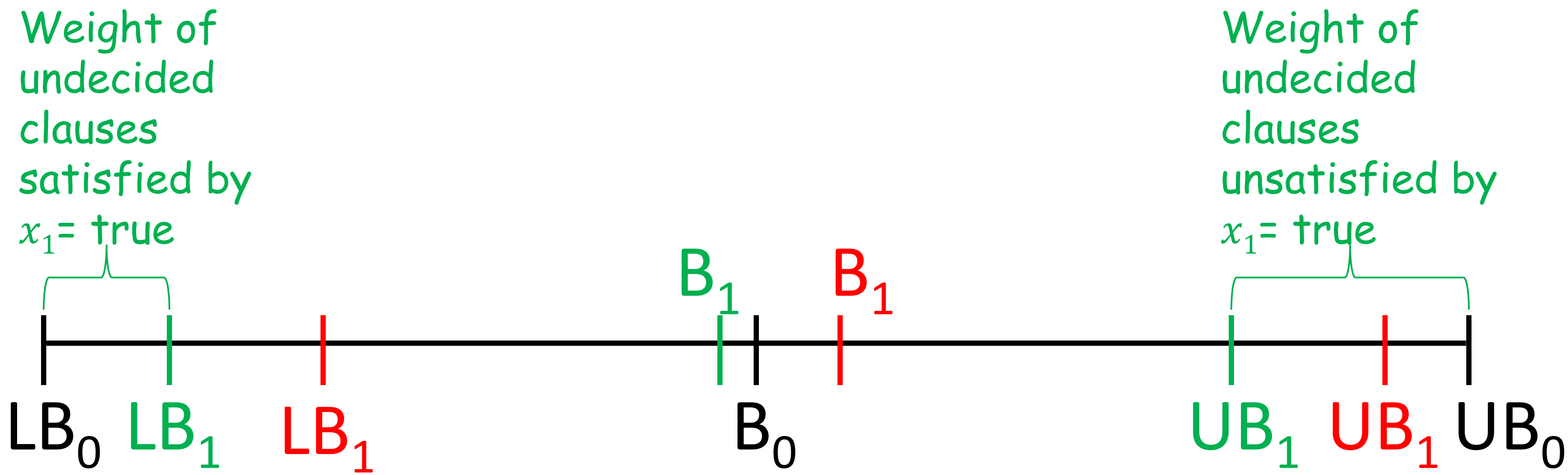
Set x_1 to true



Set x_1 to true

or

Set x_1 to false



Set x_1 to true

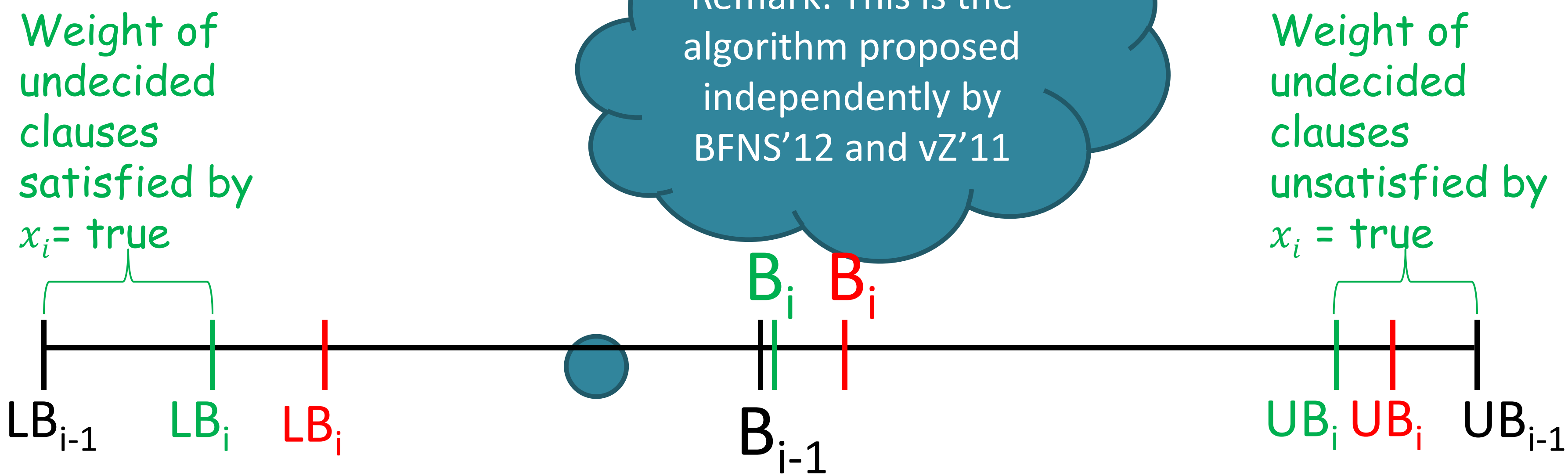
or

Set x_1 to false

Guaranteed that

$$\underbrace{(B_1\text{-}B_0)}_{t_1}+\underbrace{(B_1\text{-}B_0)}_{f_1} \geq 0$$

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



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

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

Initalize:

- LB = 0
- UB = 6

Step 1:

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

Example

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

Step 2:

- $t_2 = \frac{1}{2}(\vartriangle LB + \vartriangle UB) = \frac{1}{2}(1+0) = \frac{1}{2}$
- $f_2 = \frac{1}{2}(\vartriangle LB + \vartriangle UB) = \frac{1}{2}(3+(-1)) = 1$
- 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:

$x_1 =$ false

$x_2 =$ true w.p. 1/3 and false w.p. 2/3

$x_3 =$ true

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

Different Languages

- Bill, Baruch, and I would say:

Let G be a graph...

- Alan would say:

Let A be a matrix...

And we would be talking about the same thing!

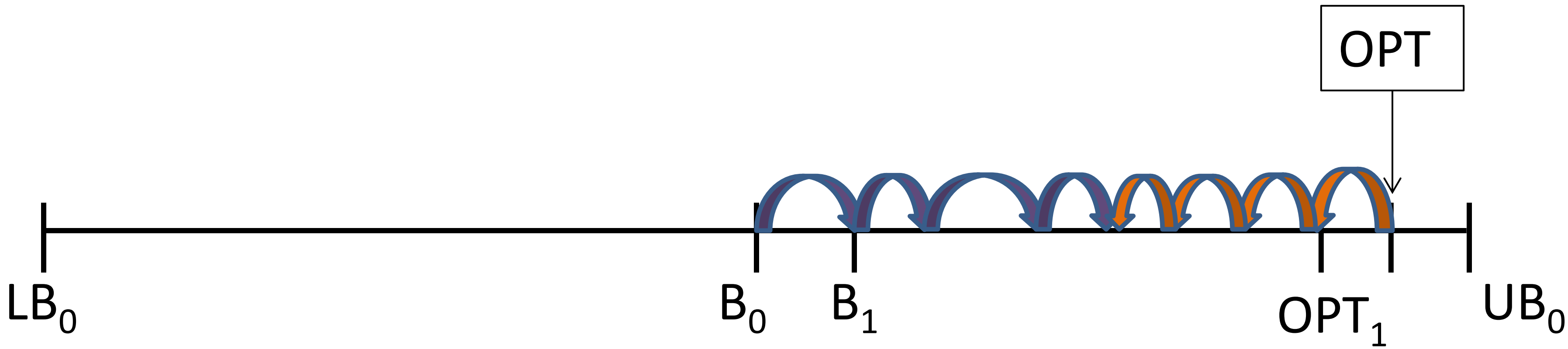
Relating Algorithm to Optimum

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

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

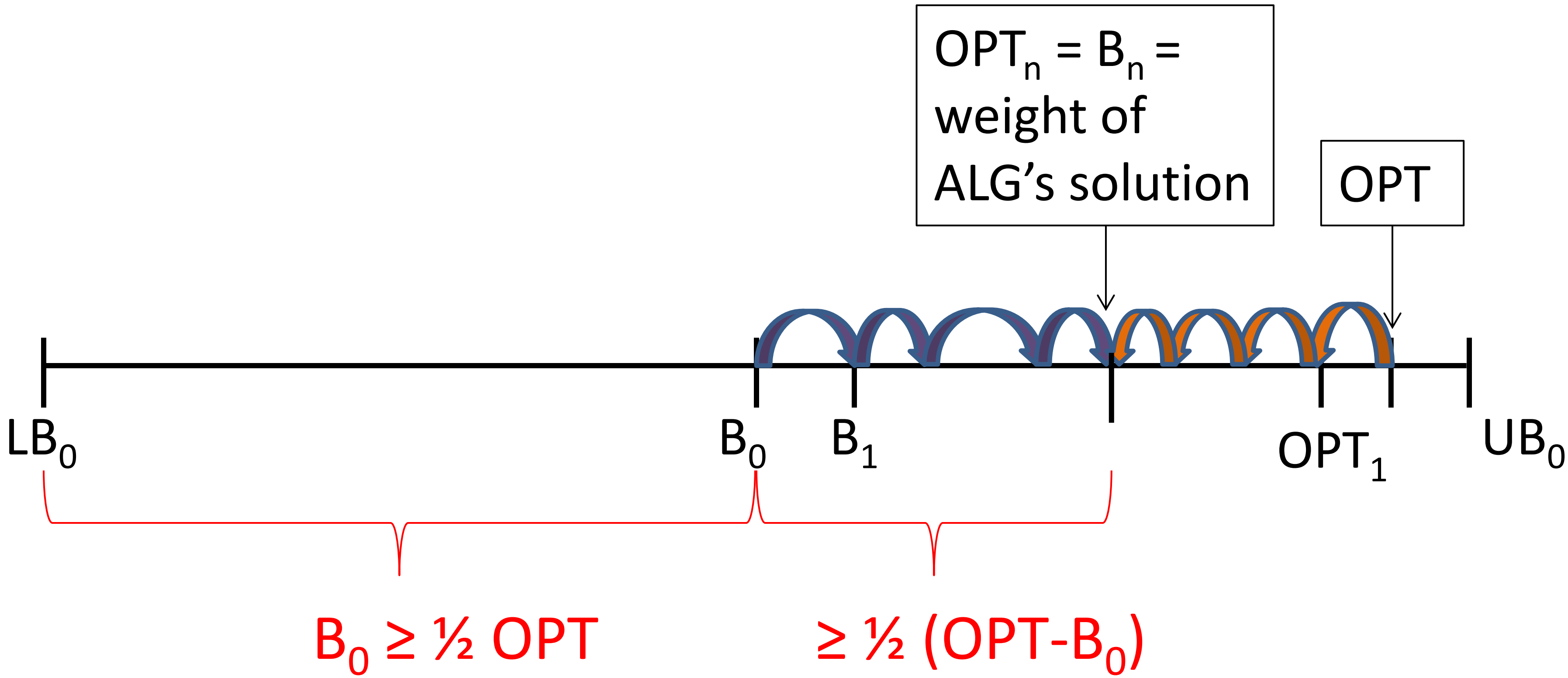
Key Lemma:

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



Key Lemma:

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

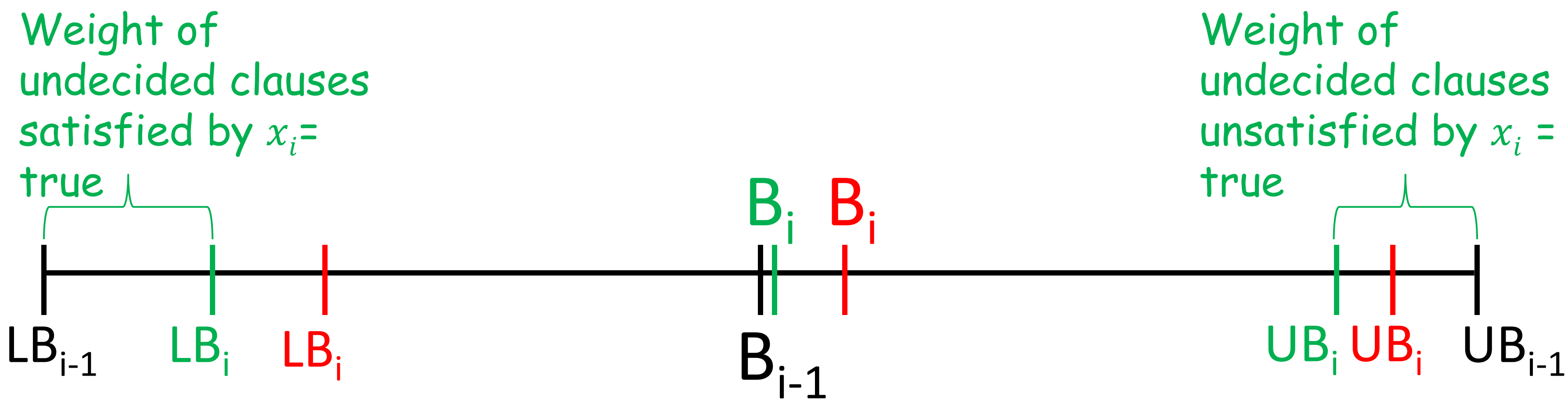


Key Lemma:

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^* = 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:

<u>Key Lemma</u>:

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

Know:

If algorithm sets x_i to true,

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

lse,

Equal to

$(t_i - f_i)^2 + 2t_i f_i$

t_i

Case 3: $t_i \geq 0$, $f_i \geq 0$ (algorithm sets x_i to tru w.p. ${}^{t_i}/_{t_i+f_i}$):

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

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

Email

Hi David,

After seeing your email, the very next thing I did this morning was to read a paper I'd earmarked from the end of the day yesterday:

Walter Gander, Gene H. Golub, Urs von Matt
"A constrained eigenvalue problem"
Linear Algebra and its Applications, vol. 114–115, March–April 1989, Pages 815–839.
"Special Issue Dedicated to Alan J. Hoffman On The Occasion Of His 65th Birthday"

The table of contents of that special issue:
http://www.sciencedirect.com.proxy.library.cornell.edu/science/journal/00243795/114/supp/C

Citations for papers in this issue:
.....

Johan Ugander

Question

Is there a simple combinatorial <u>deterministic</u> ¾-approximation algorithm?

Deterministic variant??

Greedily maximizing B_i is not good enough:

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

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

Greedily increasing B_i sets variables $x_1, \ldots, x_{n-1}$ to false
GREEDY= (n-1)(2+ε)

A negative result

Poloczek ‘11: No deterministic “priority algorithm” can be a ¾ -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 (Joint work with 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 expected fractional solution
  - **Lower bound LB** = weight of clauses already satisfied
  - **Upper bound UB** = weight of clauses not yet unsatisfied

expected

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

but either choice has bad examp

expected

- Key idea: make choices to increase **B =** ½ (**LB**+**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.

<u>Algorithm</u>:

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

Analysis

- Proof that after the first pass $E[W] \geq \frac{3}{4}\ OPT$ is identical to before.
- Proof that final solution output has value at least $E[W] \geq \frac{3}{4}\ OPT$ is via method of conditional expectation.

Conclusion

- We show this two-pass idea works for other problems as well (e.g. deterministic ½-approximation algorithm for MAX DICUT).
- Can we characterize the problems for which it does work?

Thank you for your attention

and

Happy Birthday Alan!