

CSL 857: Randomized Algorithms

Ragesh Jaiswal
CSE, IIT Delhi

Constructive LLL

Constructive LLL

- Given a k -CNF formula such that the frequency of each variable is at most $2^{\alpha k}$, we saw that there is an efficient algorithm that finds a satisfying assignment for the formula.
- On the other hand, from the non-constructive LLL, we know that any boolean formula has a satisfying assignment even when the frequency bound is as large as $\frac{2^k}{ek}$.
- Question: Is it possible to close this gap?

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
- Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
- Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)

- Let t denote the total number of calls to Fix().
- We need to upper bound t .
- Claim 1: The length of the random string R used by the algorithm is $(n + tk)$.
- Claim 2 (informal): With high probability, any description of R will need at least $(n + tk - m)$ bits.
- Question: Is there another way of encoding R ?

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
 - Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)
-
- Question: Is there another way of encoding R ?
 - Claim 3: If we are given the sequence of clauses that are called with Fix() and the final satisfying assignment, then we can figure out the string R .
 - Question: Is there a better way to encode R ?

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
- Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)

- Question: Is there another way of encoding R ?
- Claim 3: If we are given the sequence of clauses that are called with Fix() and the final satisfying assignment, then we can figure out the string R .
- Question: Is there a better way to encode R ?
- Suppose we note down the clauses that are called to fix from Solve(). We need a better way to encode the clauses within recursive calls to Fix().

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
 - Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)
-
- Suppose we note down the clauses that are called to fix from Solve(). We need a better way to encode the clauses within recursive calls to Fix().
 - Claim 4: If we are given the sequence of clauses that are called to Fix from Solve(), the “edge number” for each recursive call, some symbol denoting the end of recursive call, and the final assignment, then we can figure out R .

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
 - Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)
-
- Claim 4: If we are given the sequence of clauses that are called to Fix from Solve(), the “edge number” for each recursive call, some symbol denoting the end of recursive call, and the final assignment, then we can figure out R .
 - Question: What is the length of the above description?

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
 - Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)
-
- Claim 4: If we are given the sequence of clauses that are called to Fix from Solve(), the “edge number” for each recursive call, some symbol denoting the end of recursive call, and the final assignment, then we can figure out R .
 - Question: What is the length of the above description?
 - $m + t \cdot (\log d + O(1)) + n$ bits.

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
- Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)

- Claim 4: If we are given the sequence of clauses that are called to Fix from Solve(), the “edge number” for each recursive call, some symbol denoting the end of recursive call, and the final assignment, then we can figure out R .
- Question: What is the length of the above description?
 - $m + t \cdot (\log d + O(1)) + n$ bits.
- So we have (whp):
$$m + t \cdot (\log d + O(1)) + n \geq n + tk - m.$$

Constructive LLL

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause
 - Fix($\mathcal{C}$)
- Fix($\mathcal{C}$)
 - Randomly resample the variables of $\mathcal{C}$.
 - While there is a clause D that shares variables with $\mathcal{C}$ such that D is unsatisfied
 - Fix(D)

- So we have (whp):

$$m + t \cdot (\log d + O(1)) + n \geq n + tk - m.$$

- Which gives us:

$$2m \geq t \cdot (k - \log d - O(1)).$$

- So when $d \leq 2^{k-O(1)}$, then $t \leq m$, which implies that the algorithm is efficient.

End
