Moser's Algorithmic LLL

Presented by Zhiwei Zhang
Sep 6th, 2018

k-CNF

- conjunctive normal form:

  k-CNF $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$

- m clauses, n Boolean variables

- each clause $C_i = l_{i_1} \vee l_{v_2} \vee \cdots \vee l_{i_k}$

  is a disjunction of exact k literals

- degree d: each clause shares variables with at most d other clauses

Lovasz Local Lemma
(symmetric version)

"bad events": $B_1, B_2, \ldots, B_n$

each one happens with probability p and
is dependent on at most d other events

if $4pd \leq 1$

then

$$Pr[\overline{B_1} \wedge \overline{B_2} \wedge \cdots \wedge \overline{B_n} > 0]$$

k-CNF

- conjunctive normal form:

  k-CNF $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$

- m clauses, n Boolean variables

- each clause $C_i = l_{i_1} \vee l_{v_2} \vee \cdots \vee l_{i_k}$

  is a disjunction of exact k literals

- degree d: each clause shares variables with at most d other clauses

  What if $d \leq 2^{k-2}$?

What if $d \leq 2^{k-2}$?

for each clause $C_i = l_{i_1} \vee l_{v_2} \vee \cdots \vee l_{i_k}$

Bad event: $C_i \, is \, unsatisfied$

$$p = Pr[C_i \, is \, unsatisfied] = 2^{-k}$$

What if $d \leq 2^{k-2}$?

for each clause $C_i = l_{i_1} \vee l_{v_2} \vee \cdots \vee l_{i_k}$

Bad event: $C_i \, is \, unsatisfied$

$$p = Pr[C_i \, is \, unsatisfied] = 2^{-k}$$

$d \leq 2^{k-2}$ $\Longleftrightarrow$ $4pd \leq 1$ (LLL condition)

$\Rightarrow$ $Pr[\phi \, is \, satified] > 0$

What if $d \leq 2^{k-2}$?

for each clause $C_i = l_{i_1} \vee l_{v_2} \vee \cdots \vee l_{i_k}$

Bad event: $C_i \, is \, unsatisfied$

$$p = Pr[C_i \, is \, unsatisfied] = 2^{-k}$$

$$d \leq 2^{k-2} \iff 4pd \leq 1$$

(LLL condition)

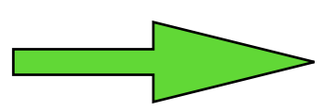
$$\rightarrow \quad Pr[\phi \, is \, satified] > 0$$

$$\rightarrow \quad \exists \text{ satisfying assignment for } \phi$$

Algorithmic LLL

ϕ : k-CNF of max degree d with m clauses on n variables

Theorem

$d \leq 2^{k-2}$  $\exists$ satisfying assignment for ϕ

Algorithmic LLL

ϕ : k-CNF of max degree d with m clauses on n variables

Theorem

$d \leq 2^{k-2}$ ⟶ $\exists$ satisfying assignment for ϕ

Theorem (Moser, 2009)

$d \leq 2^{k-3}$ ⟶ satisfying assignment can be found in $O(n + kmlogm)$ w.h.p.

ϕ : k-CNF of max degree d with m clauses on n variables

Solve(ϕ)

pick a random assignment
$x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
Fix(D)

ϕ : k-CNF of max degree d with m clauses on n variables

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

Observation: In Solve(), for every clause C, Fix(C) can be called for only once.

ϕ : k-CNF of max degree d with m clauses on n variables

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

Observation: In Solve(), for every clause C, Fix(C) can be called for only once.

# of top-level calls to Fix(C): $\leq$ m (# of clauses)

total # of calls to Fix(C) (including recursive calls): t

ϕ : k-CNF of max degree d with m clauses on n variables

Solve(ϕ)

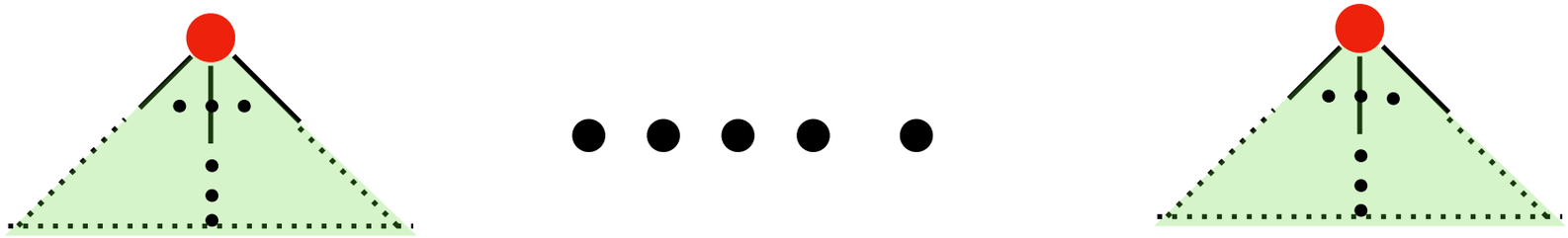
pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$\leq$ m recursion trees

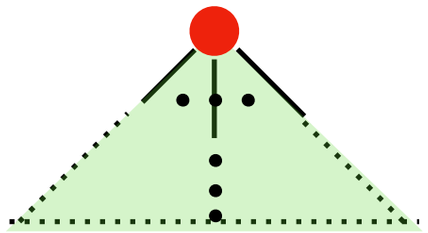
total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

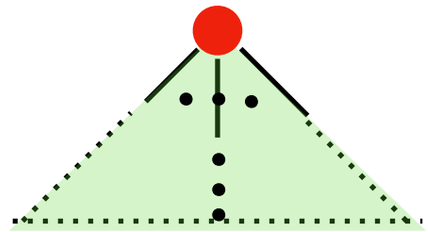
Encode the recursion trees/running log

$\leq$ m recursion trees



• • • • •

total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

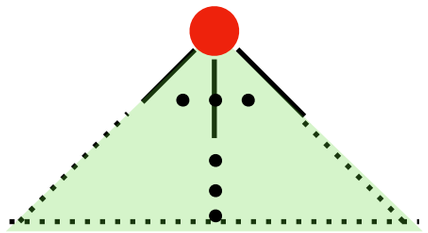
the recursion trees can be encoded to:

for each recursion tree:

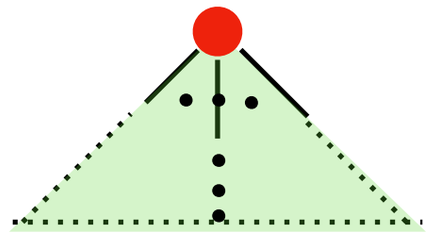
root: $\lceil log_2 m \rceil$ bits

Encode the recursion trees/running log

$\leq$ m recursion trees



total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

the recursion trees can be encoded to:

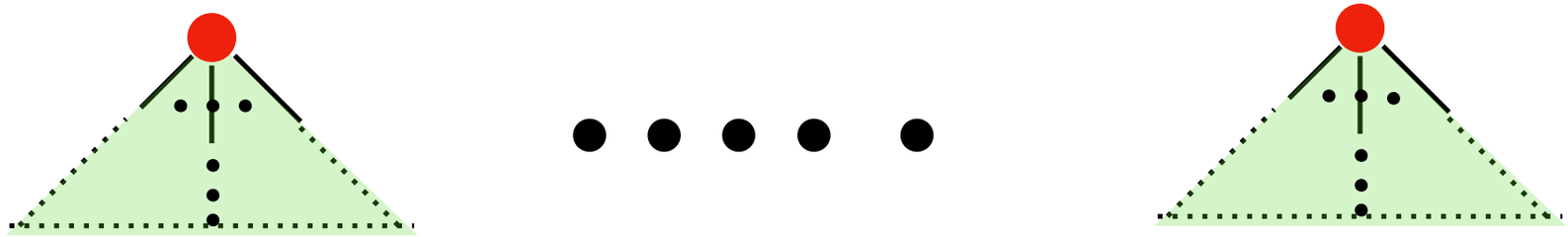
for each recursion tree:

root: $\lceil log_2 m \rceil$ bits

each internal node: $log_2 d + O(1)$ bits

Encode the recursion trees/running log

$\leq$ recursion trees total # nodes:



total # of random bits: $n + tk$ (assigned bits)

the recursion trees can be encoded to:

$$m\lceil log_2 m \rceil + t\big(log_2 d + O(1)\big)$$

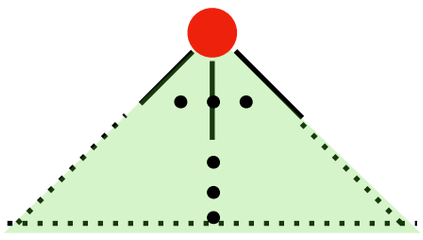
for each recursion tree:

root: bits

each internal node: bits

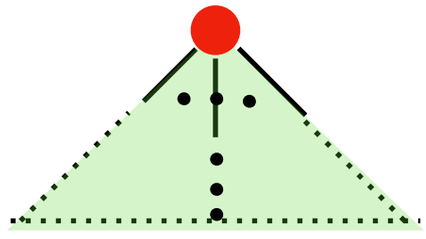
Recover the sequence of random bits

$\leq$ m recursion trees



• • • • •

total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

the sequence of random bits can be recovered from:

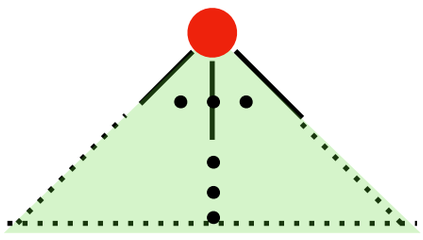
final assignment: n bits

+

recursion trees /running log: $\leq$m $\lceil log_2 m \rceil$ + t$(log_2 d + O(1))$ bits

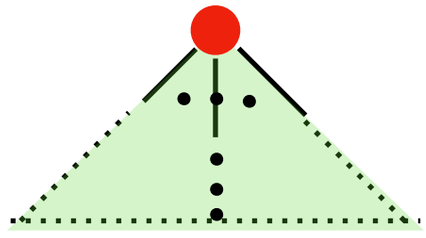
Recover the sequence of random bits

$\leq$ m recursion trees



• • • • •

total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

the sequence of random bits can be recovered from:

final assignment: n bits

+

recursion trees /running log: $\leq$m $\lceil log_2 m \rceil$ + t$(log_2 d + O(1))$ bits

Observation: Fix(C) is called → assignment of C is uniquely determined

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log
start
Fix(A)
Fix(B)
Fix(C)
end

0 1 0 1
Solution

Random Bits Assigned in Fix()

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log
start
Fix(A)
Fix(B)
Fix(C)
end

0 1 0 1
Current Assignment

011
Random Bits Assigned in Fix()

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log
start
Fix(A)
Fix(B)
Fix(C)
end

0 1 0 0
0 1 0 1
Current Assignment

011
Random Bits Assigned in Fix()

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log	Current Assignment	Random Bits Assigned in Fix()
start	0 1 0 0	100
Fix(A)	0 1 0 1	011
Fix(B)		
Fix(C)		
end		

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log	Current Assignment	Random Bits Assigned in Fix()
start	0 1 0 0	
Fix(A)	0 0 0 0	000
Fix(B)	0 1 0 1	011
Fix(C)		
end		

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

$$A = \neg x_1 \vee x_2 \vee x_3$$
$$B = \neg x_2 \vee x_3 \vee x_4$$
$$C = x_1 \vee x_2 \vee x_4$$

$$\phi = A \wedge B \wedge C$$

$$\phi(0, 1, 0, 1) = 1$$

Log	Current Assignment	Random Bits Assigned in Fix()
start	1 0 0 0	
Fix(A)	0 1 0 0	010
Fix(B)	0 0 0 0	000
Fix(C)	0 1 0 1	011
end		

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

Log
start
Fix(A)
Fix(B)
Fix(C)
end

0 1 0 1

Log+ Final Solution

Recover

1 0 0 0
010
000
011

All random bits used in Solve() and Fix()

Demonstration

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

Log
start
Fix(A)
Fix(B)
Fix(C)
end

0 1 0 1

Log+ Final Solution

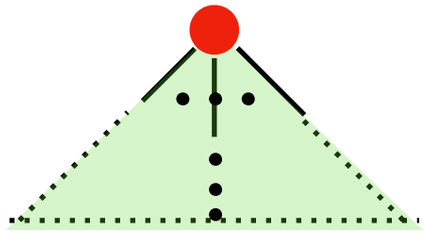
Recover

Encode

1 0 0 0
010
000
011

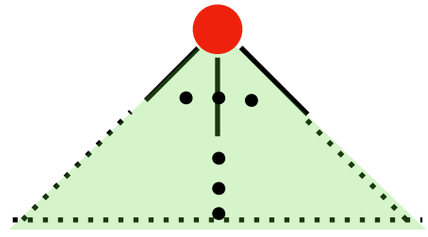
All random bits used in Solve() and Fix()

$\leq$ m recursion trees



• • • • •

total # nodes: t

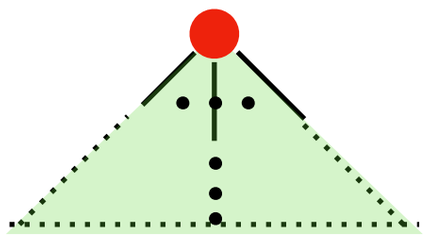


total # of random bits: $n + tk$ (assigned bits)

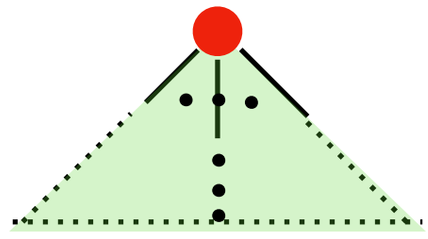
All random bits used in Solve() and Fix() is encoded to:

Log+ Final Solution: $n + m\lceil log_2 m \rceil + t(log_2 d + O(1))$ bits

$\leq$ m recursion trees



total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

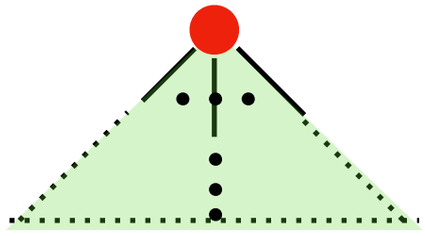
the sequence of random bits is encoded to:

$n +$ $m\lceil log_2 m \rceil + t(log_2 d + O(1))$ bits

Incompressibility Theorem (Kolmogorov)

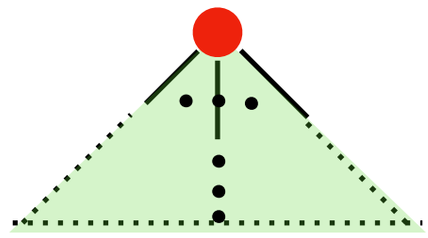
N uniform random bits cannot be encoded to less than N-l bits with probability $1 - O(2^{-l})$

$\leq$ m recursion trees



• • • • •

total # nodes: t



total # of random bits: (assigned bits)

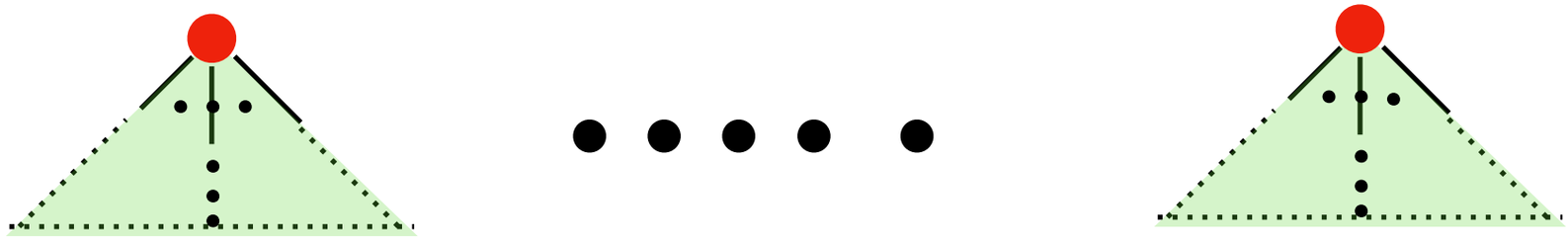
the sequence of random bits is encoded to:

bits

$$n + tk \leq \; n + \; m\lceil log_2 m \rceil + t(log_2 d + O(1)) \text{ w.h.p.}$$

$\leq$ m recursion trees

total # nodes: t



total # of random bits: $n + tk$ (assigned bits)

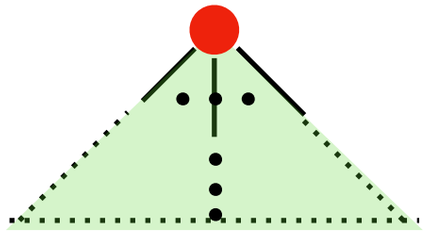
the sequence of random bits is encoded to:

$$n + \quad m\lceil log_2 m \rceil + t(log_2 d + O(1)) \text{ bits}$$

$$n + tk \;\leq\; n + \quad m\lceil log_2 m \rceil + t(log_2 d + O(1)) \text{ w.h.p.}$$

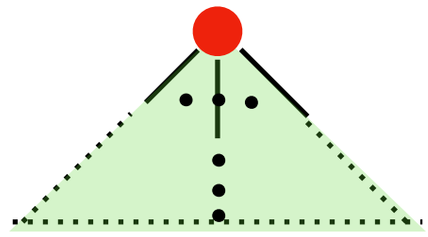
when $d < 2^{k-c}$ $\longrightarrow$ $t \leq \dfrac{m log_2 m}{c - O(1)}$

$\leq$ m recursion trees



• • • • •

total # nodes: t



total # of random bits: (assigned bits)

the sequence of random bits is encoded to:

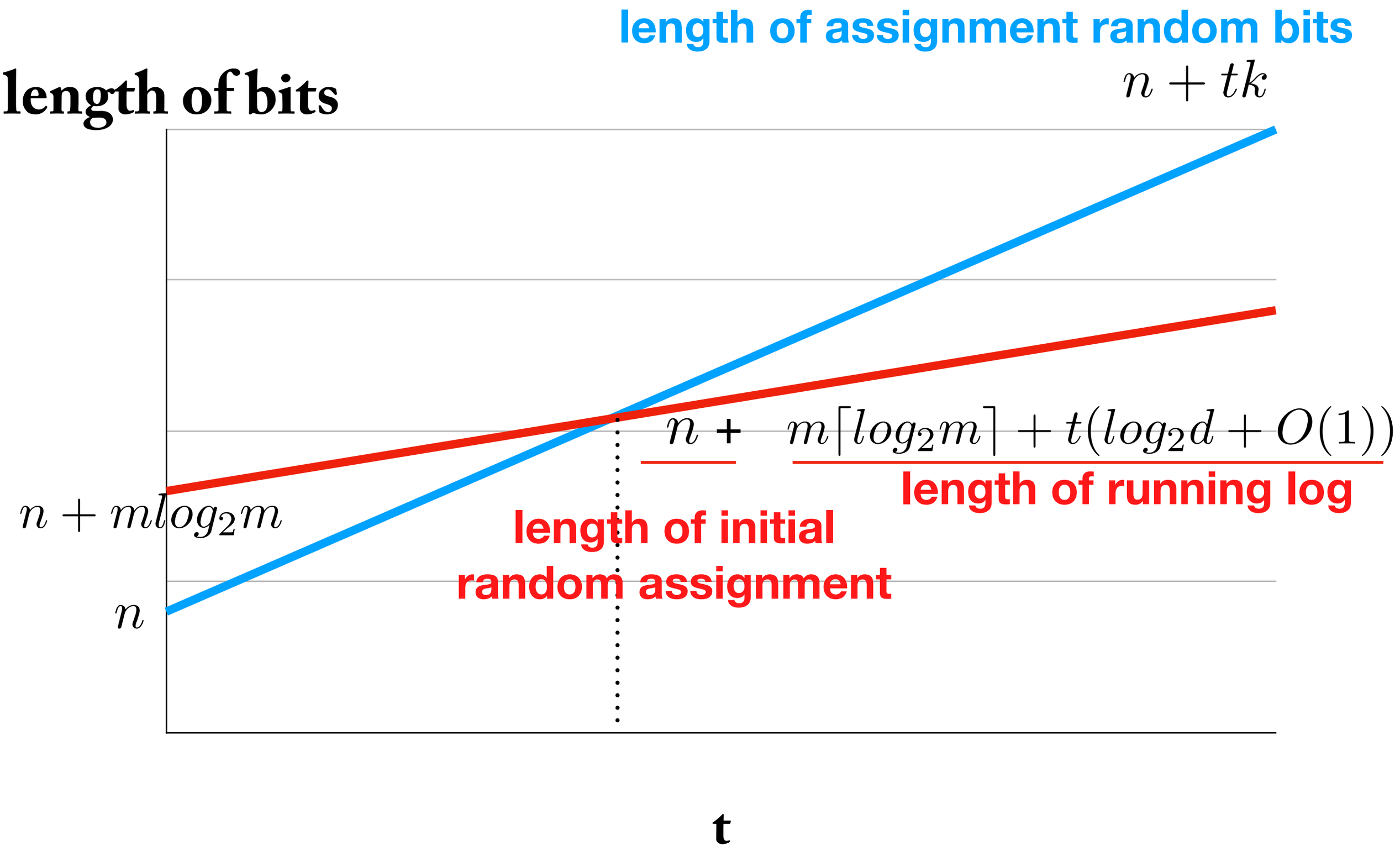
$$n + \quad m\lceil log_2 m\rceil + t(log_2 d + O(1)) \text{ bits}$$

$$n + tk \;\leq\; n + \quad m\lceil log_2 m\rceil + t(log_2 d + O(1)) \text{ w.h.p.}$$

when $d < 2^{k-c}$ → $t \leq \frac{m log_2 m}{c - O(1)}$

total running time(total assigned random bits):

$$n + tk \;= O(n + kmlog_2 m)$$


length of assignment random bits
length of bits
$n + tk$
$n + m\lceil log_2 m \rceil + t(log_2 d + O(1))$
length of running log
$n + mlog_2 m$
length of initial
random assignment
n
t

length of assignment random bits

length of bits

$n + tk$

$n + m\lceil log_2 m \rceil + t(log_2 d + O(1))$

length of running log

$n + mlog_2 m$

length of initial
random assignment

n

w.h.p

almost impossible to happen!

t

length of assignment random bits

length of bits

$n + tk$

$n + \quad m\lceil log_2 m \rceil + t(log_2 d + O(1))$

length of running log

$n + mlog_2 m$

length of initial
random assignment

n

w.h.p

almost impossible to happen!

$O(n + kmlog_2 m)$

t

Algorithmic LLL

ϕ : k-CNF of max degree d with m clauses on n variables

Theorem (Moser, 2009)

$d \leq 2^{k-3}$ → satisfying assignment can be found in $O(n + kmlogm)$ w.h.p.

Solve(ϕ)

pick a random assignment $x_1, x_2, \ldots, x_n$;
while exists unsatisfied clause C
  Fix(C)

Fix(C)

replace variables in C with random values
while exists unsatisfied clause D overlapping with C
  Fix(D)

Acknowledgement

Prof. Yitong Yin, Nanjing University

A majority of pages of this slides are made using reference of Prof. Yin slides and materials in class

Reference

[1]A constructive Proof of the Lovasz Local Lemma, Robin A.Moser

[2]https://terrytao.wordpress.com/2009/08/05/mosers-entropy-compression-argument/

[3]http://tcs.nju.edu.cn/slides/aa2017/LLL.pdf