A new paradigm for k-SAT

Evgeny DANTSIN
Edward HIRSCH
Steklov Mathematical Institute
Russian Academy of Sciences

Presented by Ioannis Koutis and Ke Yang

April 11, 2000

Carnegie Mellon

A deterministic algorithm

- Best deterministic time algorithm for
  3-SAT: $poly(n)1.497^n$ [Kul,98]
  k-SAT: $poly(n)2^{(1-(1/2k))n}$, [PaPuZa,97]
- Given $P \neq NP$ we seek $1 < c \leq 1.497$
- Very complicated results!
- Schoening's Derandomization.
  A new paradigm. A new start!
- Wonderful connection with Coding Theory

Carnegie Mellon

Schoening's algorithm overview

1	We start with a random assignment
2	If satisfiying, okay we are done
3	Otherwise we pick an arbitrary unsatisfied clause and randomly flip a literal.
4	Repeat step 3 for $3n$ times, and if we still fail to find the solution, go back to step 1

Must derandomize two features:

1. Random Initial Assignment A
2. Random Walk starting from A

Initial assignment selection

Key Feature (w.h.p.) :
"Quickly" find an assignment close to S.

What could be a derandomization?

Answer:
Deterministically construct a set C of assignments which contains at least one "good" assignment.

Critical Issues:

1. Size of C
2. Miminum distance of C to S

Carnegie Mellon

Coding Theory Excursion

Key Idea :
Think of assignments as binary words....

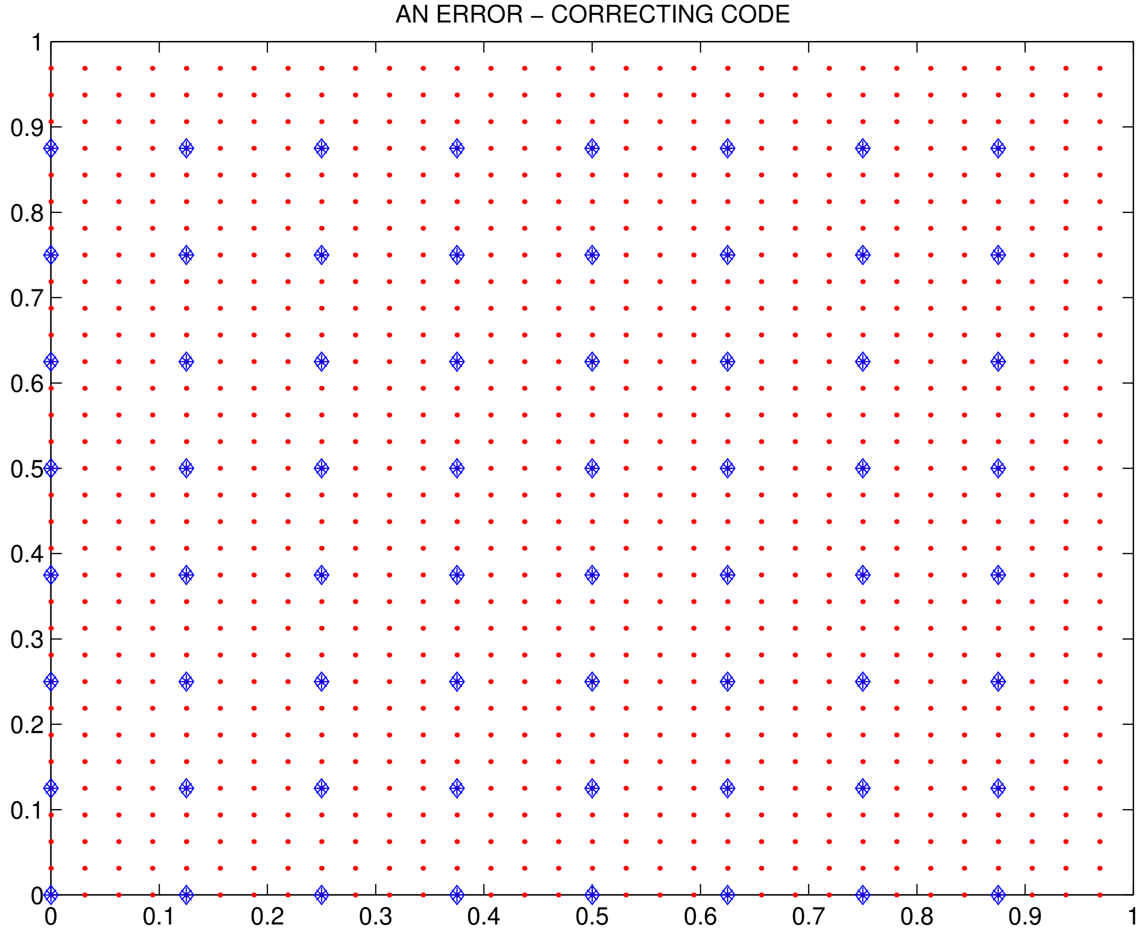
What is an covering code?

- Set of words $C = \{w_1, w_2, \ldots\}$
- $|C| < 2^n$
- $distance(w_i, w_j) < 2R$

All words are to distance $< R$ from C

Visualization

Cover the 2^n binary words with some "small" number of codewords...



Intuition: Find an even distribution....

Technical Details

$H(x) = -x \log_2(x) - (1-x)\log_2(1-x)$

Basic Lemma: For every $\epsilon > 0$, there exists a code C with

- $|C| = 2^{(1-H(\frac{1}{k+1})+\epsilon)n}$
- $R = \lfloor (\frac{1}{k+1} + \epsilon)n \rfloor$
- C can be generated in amortized polynomial time.

For $k = 3$ this gives

- $R = (0.25 + \epsilon')n$
- $|C| = 2^{(0.1887+\epsilon)n} = 1.139^n$
- Compare with Shcoening's $(1/1.05^n)$.

Carnegie Mellon

Random Walks

- How can we derandomize random flippings of variables?
- Try all possible random walks....
- But, a random walk can be very long!

Key Observation: Use 1^{st} derandomization
We are at distance R from the satisfying S.

- Follow all possible random walks of length $< R$
- Don't flip twice the same variable in a walk
- This defines a k-ary tree of depth R...
- and size k^R

Carnegie Mellon

Putting together

Generate code $C = w_1, \ldots, w_{|C|}$
for $i = 1, \ldots, |C|$ **do**
    Start in w_i
    Depth first search in k-ary tree
    if S found **return** S
end for
return "not satisfied"

Soundness "proof". By definition of C we peek a "good" starting point A, and the tree with root A contains S.

Overall Complexity:
$time\_for\_C \times time\_for\_tree = |C| \times k^R = poly(n)(\frac{2k}{k+1} + \epsilon)^n$

1.5^n for 3-SAT!!

Carnegie Mellon

Improved Bound for 3-SAT

- Use structure of unsatisfied clauses
- Flip more than one variables ...
- Give an example here ..
- But this creates unbalanced trees, and a node can have more than 3 sons..

Question:

How can we estimate the number of leaves in an unbalanced tree?

A nice general result...

Carnegie Mellon

Using Structure

- Pick an unsatisfied clause $\{a_1, a_2, a_3\}$
- Flip a_1
- A clause of the form $\{b_1, b_2\}$ or $\{b_1\}$ or $\{\}$ is produced
- We can flip together $\{a_1, b_1\}$ or $\{a_1, b_2\}$

Carnegie Mellon

Branching Vectors

[Kullman,Luckhardt,97]

Some Definitions !

- A tree with root r
- A node u with sons $u_1, \ldots, u_s$
- Height of u, $h(u)$ (maximal distance from a leaf)
- A branching vector of u, $\theta = (\theta_1, \ldots, \theta_n)$, where $0 < \theta_i \leq h(u) - h(u_i)$
- The characteristic polynomial $1 - \Sigma_{i=1}^{s} x^{-\theta_i}$
- The root is the branching number $b(u)$

Carnegie Mellon

Branching Vectors

Two examples

- The (1,1,1) case
- The (1,2) case ...golden ratio

Theorem

- $b_{\mathsf{max}} = \max_u b(u)$
- Number of leaves $\leq (b_{\mathsf{max}})^{h(r)}$
- Number of nodes $\leq h(r)(b_{\mathsf{max}})^{h(r)}$

Carnegie Mellon

Improved Bound for 3-SAT

- A careful 2-step look-ahead analysis..
- 5 cases...
- Worst case branching vector (1,1,2,2,3,3)
- Worst case branching number $\mu = b_{\text{max}} < 2.9197$

Choice of appropriate code gives the

Final Result : 3-SAT is decidable in time

$$p(n)(\frac{2\mu}{\mu+1} + \epsilon)^n < p(n)(1.4898 + \epsilon)^n$$

General Discussion

- Generalization to CSP's.
- We just try all valid changes in the values of variables
- In general, larger trees..
- But still good algorithms comparing to exhaustive search.

- Improvement 1: Better covering codes?
- Improvement 2: Smaller trees?
- Improvement 3: What else??

Carnegie Mellon