Randomized Algorithms

Randomized 2-SAT and 3-SAT

Joseph Chuang-Chieh Lin

Department of Computer Science & Engineering
National Taiwan Ocean University

Spring 2026

Outline

Roadmap

- Start with SAT as a search problem on truth assignments.
- A simple randomized local-search algorithm for 2-SAT.
- Analyze its progress using a one-dimensional random walk.
- Reuse the same idea for 3-SAT, where the walk becomes biased in the wrong direction.
- Fix the issue with random restarts and get a faster-than-brute-force randomized algorithm.


Boolean formulas in CNF

Conjunctive Normal Form (CNF)

A literal is a variable x_i or its negation $\neg x_i$.

A clause is an OR (i.e., $\vee$) of literals.

A CNF formula is an AND of clauses.

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3)$$

CNF formula: AND of clauses

k-SAT

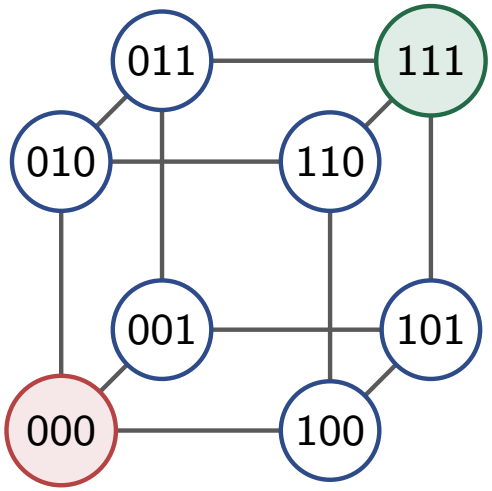
- In k-SAT, each clause has exactly k literals.
- 2-SAT: each clause has two literals.
- 3-SAT: each clause has three literals.
- General SAT is NP-hard; 3-SAT is NP-complete.
- 2-SAT is polynomial-time solvable, but the randomized analysis is useful because it introduces the Markov-chain method.

Example: a 2-SAT formula

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$$


Truth assignments as a hypercube

- A truth assignment is a bit string in $\{0,1\}^n$.
- Flipping one variable moves to a neighboring vertex of the hypercube.
- Local search is a random walk on this assignment graph.



assignments form the n-dimensional hypercube

The local-search template

Randomized repair step

1. If all clauses are satisfied, stop.
2. Pick an unsatisfied clause.
3. Choose one literal in that clause uniformly at random.
4. Flip its variable.

- The clause becomes satisfied immediately after the flip.
- But other clauses may become unsatisfied.
- The analysis tracks progress toward one fixed satisfying assignment S.

The local-search template

Randomized repair step

1. If all clauses are satisfied, stop.
2. Pick an unsatisfied clause.
3. Choose one literal in that clause uniformly at random.
4. Flip its variable.

- The clause becomes satisfied immediately after the flip.
- But other clauses may become unsatisfied.
- The analysis tracks progress toward one fixed satisfying assignment S.
  - There could exist more than one satisfying assignment.

Why Markov chains enter

- The full assignment process has 2^n possible states.
- For the analysis, collapse the state to a single number:

$$X_i = \#\{\text{variables where current assignment } A_i \text{ agrees with } S\}.$$

- If $X_i = n$, the current assignment equals S and is satisfying.
- The key question becomes: How long until this one-dimensional process hits n?

X_i = number of matches

Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.


Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.
- Suppose the current assignment is $A_i = (0, 1, 0)$.


Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.
- Suppose the current assignment is $A_i = (0, 1, 0)$.
- Then A_i agrees with S only on x_2, so $X_i = 1$.


Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.
- Suppose the current assignment is $A_i = (0, 1, 0)$.
- Then A_i agrees with S only on x_2, so $X_i = 1$.
- Under A_i, the clause $(x_1 \vee \neg x_2)$ is unsatisfied. The algorithm flips one of its variables uniformly at random:

Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.
- Suppose the current assignment is $A_i = (0, 1, 0)$.
- Then A_i agrees with S only on x_2, so $X_i = 1$.
- Under A_i, the clause $(x_1 \vee \neg x_2)$ is unsatisfied. The algorithm flips one of its variables uniformly at random:

$$(0, 1, 0) \to \begin{cases} (1, 1, 0), & \text{if flip } x_1, \quad X_{i+1} = 2, \\ \end{cases}$$

Example

- Let
$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3),$$
- Choose a satisfying assignment $S = (1, 1, 1)$.
- Suppose the current assignment is $A_i = (0, 1, 0)$.
- Then A_i agrees with S only on x_2, so $X_i = 1$.
- Under A_i, the clause $(x_1 \vee \neg x_2)$ is unsatisfied. The algorithm flips one of its variables uniformly at random:

$$(0,1,0) \to \begin{cases} (1,1,0), & \text{if flip } x_1, \quad X_{i+1} = 2, \\ (0,0,0), & \text{if flip } x_2, \quad X_{i+1} = 0. \end{cases}$$

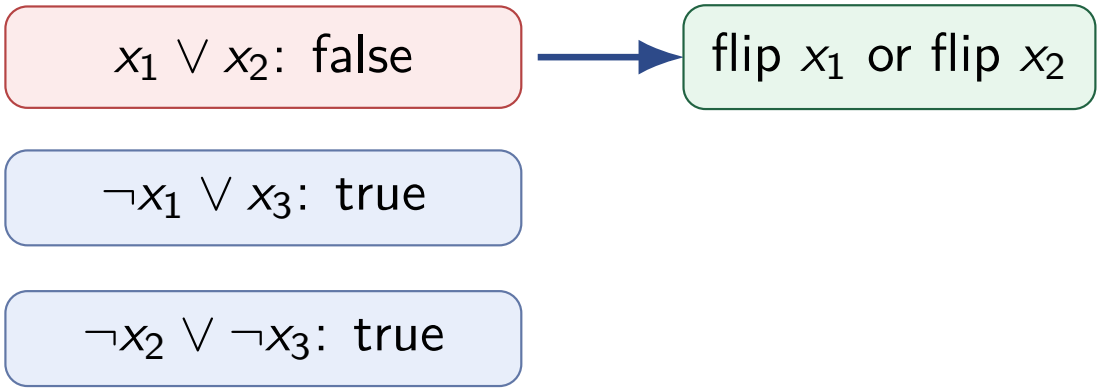

Randomized 2-SAT Algorithm

2-SAT Algorithm

1. Start with an arbitrary truth assignment.
2. Repeat up to $2mn^2$ times:
   1. if all clauses are satisfied, stop;
   2. choose an arbitrary unsatisfied clause;
   3. choose one of its two literals uniformly at random;
   4. flip the chosen variable.
3. If a satisfying assignment was found, return it.
4. Otherwise return "unsatisfiable."


Example: one repair step

$$\phi = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3)$$



- Suppose $A = (x_1, x_2, x_3) = (0, 0, 1)$.
- Clause $(x_1 \vee x_2)$ is unsatisfied.
- The algorithm flips either x_1 or x_2, each with probability $1/2$.

Fix a satisfying assignment

- Assume the formula is satisfiable.
- Fix one satisfying assignment S.
- Let A_i be the current assignment after i flips.
- Let $X_i = \#\{\ell : A_i(x_\ell) = S(x_\ell)\}$.
- Reaching $X_i = n$ is sufficient for success.

Observation (Worst Case)

The algorithm may find another satisfying assignment before $X_i = n$; ignoring this only makes the analysis pessimistic.


Fix a satisfying assignment

- Assume the formula is satisfiable.
- Fix one satisfying assignment S.
- Let A_i be the current assignment after i flips.
- Let $X_i = \#\{\ell : A_i(x_\ell) = S(x_\ell)\}$.
- Reaching $X_i = n$ is sufficient for success.

Observation (Worst Case)

The algorithm may find another satisfying assignment before $X_i = n$; ignoring this only makes the analysis pessimistic.

$$\{A_i = S\} \subseteq \{A_i \text{ is satisfying}\}.$$

Why an unsatisfied clause helps

Key logic

Suppose clause C is unsatisfied by A_i, but S satisfies C.

- Since A_i makes every literal in C false, but S makes at least one literal in C true,
- A_i and S must disagree on at least one variable appearing in C.
- For a 2-SAT clause, there are only two variables.
- Therefore, a uniformly random flip among the two literals improves X_i with probability $\geq 1/2$.

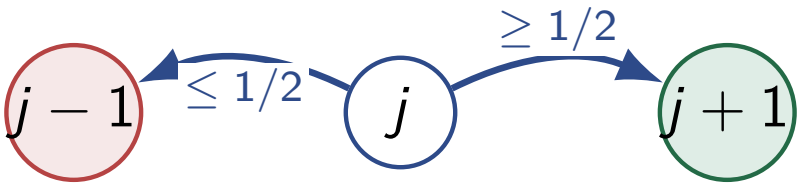

Transition inequalities for 2-SAT

For $1 \leq j \leq n-1$,

$$\Pr[X_{i+1} = j+1 \mid X_i = j] \geq \frac{1}{2},$$

and

$$\Pr[X_{i+1} = j-1 \mid X_i = j] \leq \frac{1}{2}.$$



A pessimistic Markov chain

The process X_i is not necessarily Markovian: the probability of improvement can depend on the detailed current assignment.

Pessimistic comparison chain Y_i

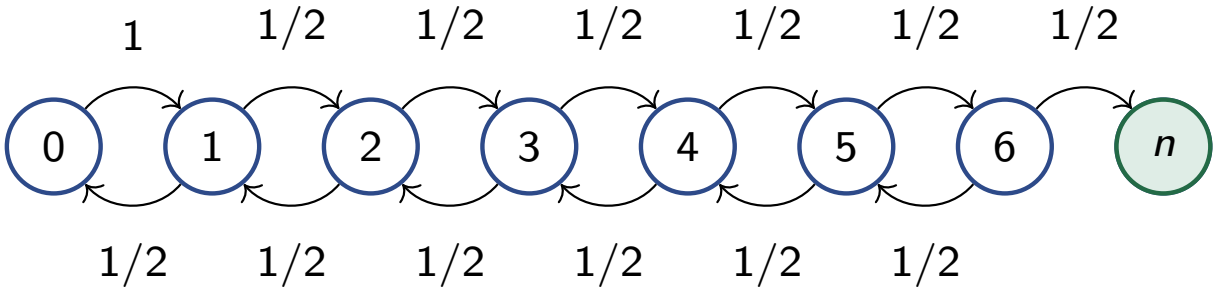
$$Y_0 = X_0, \quad \Pr[Y_{i+1} = 1 \mid Y_i = 0] = 1,$$

and for $1 \leq j \leq n-1$,

$$\Pr[Y_{i+1} = j+1 \mid Y_i = j] = \frac{1}{2}, \quad \Pr[Y_{i+1} = j-1 \mid Y_i = j] = \frac{1}{2}.$$

- Y_i is no more favorable than X_i.
- Its hitting time of n upper-bounds the expected time for the algorithm.

The random-walk picture for 2-SAT



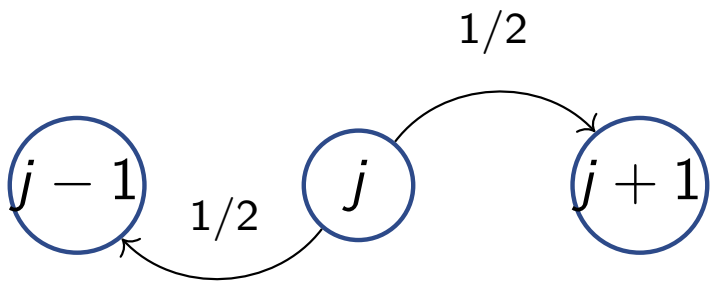
state j = number of variables matching a fixed satisfying assignment S

- This is a simple random walk on $\{0, 1, \ldots, n\}$.
- State n is the target: the current assignment equals S.
- State 0 deterministically moves to 1.

Hitting-time notation

h_j: the expected number of steps for Y_i to reach n, starting from $Y_0 = j$.

- $h_n = 0$.
- $h_0 = h_1 + 1$ because state 0 always moves to 1.
- For $1 \leq j \leq n-1$, condition on the next step.



Hitting-time equations for 2-SAT

System of equations

$$h_n = 0,$$
$$h_j = \frac{h_{j-1}}{2} + \frac{h_{j+1}}{2} + 1, \quad 1 \le j \le n-1,$$
$$h_0 = h_1 + 1.$$

- The $+1$ accounts for the step just taken.
- The average accounts for the two equally likely next states.
- These equations have a unique solution.


Solving the equations: finite differences

Claim

For $0 \leq j \leq n-1$,

$$h_j = h_{j+1} + 2j + 1.$$

- At $j = 0$, this is $h_0 = h_1 + 1$.
- The recurrence implies

$$h_{j+1} = 2h_j - h_{j-1} - 2.$$

- If $h_{j-1} = h_j + 2(j-1) + 1$, then

$$h_{j+1} = h_j - 2j - 1.$$


Expected hitting time for 2-SAT

Summing the finite differences gives

$$h_0 = h_1 + 1 = h_2 + 1 + 3 = \cdots = \sum_{i=0}^{n-1} (2i + 1) = n^2.$$

Equivalently, the solution is

$$h_j = n^2 - j^2.$$

From any starting assignment, the expected number of repair steps until matching S is at most n^2.

Lemma 1

- Assume a 2-SAT formula with n variables has a satisfying assignment and the algorithm is allowed to run until it finds one.
- Then the expected number of steps until the algorithm finds a satisfying assignment is at most n^2.

Lemma 1

- Assume a 2-SAT formula with n variables has a satisfying assignment and the algorithm is allowed to run until it finds one.
- Then the expected number of steps until the algorithm finds a satisfying assignment is at most n^2.

- The proof analyzes the pessimistic chain Y_i.
- The real algorithm can only do better, because an unsatisfied 2-clause offers an improving flip with probability at least 1/2.

From expectation to high-probability success

- Lemma 1 gives an expected hitting time of at most n^2.
- To avoid running forever, the algorithm uses m segments of length $2n^2$.
- In any segment, regardless of where it starts, let Z be the time to find a satisfying assignment.

From expectation to high-probability success

- Lemma 1 gives an expected hitting time of at most n^2.
- To avoid running forever, the algorithm uses m segments of length $2n^2$.
- In any segment, regardless of where it starts, let Z be the time to find a satisfying assignment.

By Markov's inequality,

$$\Pr[Z > 2n^2] \leq \frac{\mathbb{E}[Z]}{2n^2} \leq \frac{n^2}{2n^2} = \frac{1}{2}.$$

One-sided error

Theorem 1

The 2-SAT algorithm always returns a correct answer if the formula is unsatisfiable. If the formula is satisfiable, then with probability at least

$$1 - 2^{-m}$$

it returns a satisfying assignment.

- If it returns an assignment, the assignment is checked and correct.

One-sided error

Theorem 1

The 2-SAT algorithm always returns a correct answer if the formula is unsatisfiable. If the formula is satisfiable, then with probability at least

$$1 - 2^{-m}$$

it returns a satisfying assignment.

- If it returns an assignment, the assignment is checked and correct.
- If it says "unsatisfiable," it may be wrong only when

One-sided error

Theorem 1

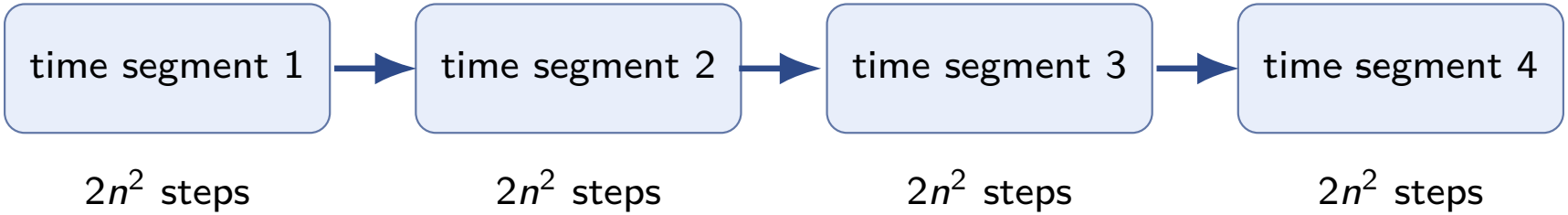
The 2-SAT algorithm always returns a correct answer if the formula is unsatisfiable. If the formula is satisfiable, then with probability at least

$$1 - 2^{-m}$$

it returns a satisfying assignment.

- If it returns an assignment, the assignment is checked and correct.
- If it says "unsatisfiable," it may be wrong only when
  - the formula was satisfiable but
  - the random walk did not find a solution in time.

Why the failure probability is 2^{-m}



Therefore,

$$\Pr[\text{fail in all } m \text{ segments}] \leq \left(\frac{1}{2}\right)^m.$$

Runtime accounting for 2-SAT

- Number of repair steps: $2mn^2$.
- A naïve implementation may scan for an unsatisfied clause in $O(n^2)$ time, since there are $O(n^2)$ distinct 2-clauses. (WHY?)

Runtime accounting for 2-SAT

- Number of repair steps: $2mn^2$.
- A naïve implementation may scan for an unsatisfied clause in $O(n^2)$ time, since there are $O(n^2)$ distinct 2-clauses. (WHY?)

Probability parameter

To get failure probability at most ε, take

$$m \geq \lg_2(1/\varepsilon).$$

Runtime accounting for 2-SAT

- Number of repair steps: $2mn^2$.
- A naïve implementation may scan for an unsatisfied clause in $O(n^2)$ time, since there are $O(n^2)$ distinct 2-clauses. (WHY?)

Probability parameter

To get failure probability at most ε, take

$$m \geq \lg_2(1/\varepsilon).$$

$$2^{-m} \leq \epsilon$$

Runtime accounting for 2-SAT

- Number of repair steps: $2mn^2$.
- A naïve implementation may scan for an unsatisfied clause in $O(n^2)$ time, since there are $O(n^2)$ distinct 2-clauses. (WHY?)

Probability parameter

To get failure probability at most ε, take

$$m \geq \lg_2(1/\varepsilon).$$

$$2^{-m} \leq \epsilon \Rightarrow m \geq \lg(1/\epsilon).$$

Takeaways from 2-SAT

1. A local search algorithm can be analyzed by tracking *distance* to one fixed solution.
2. The algorithm requires

$$\mathbb{E}[\text{remaining time to success} \mid A] \leq n^2,$$

   for each current assignment A.
3. It is enough to dominate it by a simpler pessimistic Markov chain.
4. Expected hitting time plus Markov's inequality gives a Monte Carlo algorithm with tunable one-sided error.

Moving from 2-SAT to 3-SAT

- The same repair step applies to 3-SAT: choose an unsatisfied clause and flip one of its literals uniformly at random.
- But an unsatisfied 3-clause may have only one variable on which A_i and S disagree.
- Therefore the probability of improving the match count is only guaranteed to be at least 1/3. (;;) orz

Key change/issue

$$\text{improvement probability: } \frac{1}{2} \longrightarrow \frac{1}{3}.$$

A Naïve Randomized 3-SAT

3-SAT Algorithm

1. Start with an arbitrary truth assignment.
2. Repeat up to m times:
   1. if all clauses are satisfied, stop;
   2. choose an arbitrary unsatisfied clause;
   3. choose one of its three literals uniformly at random;
   4. flip the chosen variable.
3. If a satisfying assignment was found, return it.
4. Otherwise return "unsatisfiable."

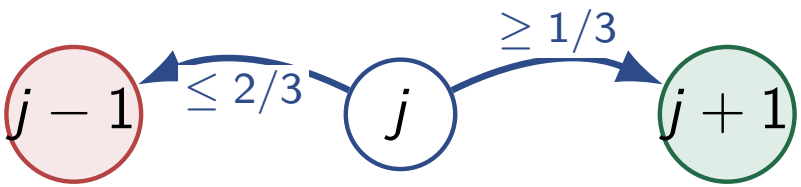

Transition inequalities for 3-SAT

Again fix a satisfying assignment S and let X_i be the number of matches with S. For $1 \leq j \leq n-1$,

$$\Pr[X_{i+1} = j+1 \mid X_i = j] \geq \frac{1}{3},$$

and

$$\Pr[X_{i+1} = j-1 \mid X_i = j] \leq \frac{2}{3}.$$



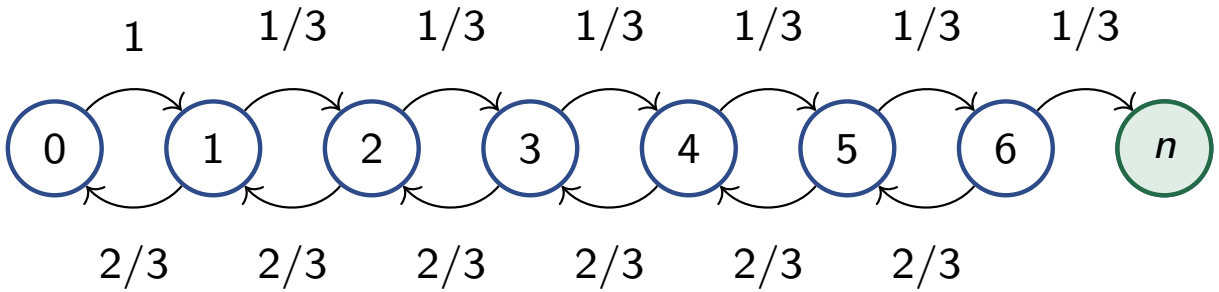
Pessimistic chain for 3-SAT

Define $Y_0 = X_0$ and

$$\Pr[Y_{i+1} = 1 \mid Y_i = 0] = 1,$$

while for $1 \le j \le n-1$,

$$\Pr[Y_{i+1} = j+1 \mid Y_i = j] = \frac{1}{3}, \qquad \Pr[Y_{i+1} = j-1 \mid Y_i = j] = \frac{2}{3}.$$



state j = number of variables matching a fixed satisfying assignment S

The drift is now unfavorable

- In 2-SAT, the pessimistic walk was unbiased.
- In 3-SAT, the pessimistic walk moves away from n with probability $2/3$.
- The target n is uphill against the drift.

Hitting-time equations for 3-SAT

Let h_j be the expected time to hit n from state j in the pessimistic chain.

Equations

$$h_n = 0,$$
$$h_j = \frac{2h_{j-1}}{3} + \frac{h_{j+1}}{3} + 1, \quad 1 \le j \le n-1,$$
$$h_0 = h_1 + 1.$$

- Same first-step analysis as before.
- Different probabilities change the solution dramatically.

Solution of the 3-SAT recurrence

The solution is

$$h_j = 2^{n+2} - 2^{j+2} - 3(n-j).$$

Equivalently, one can prove by induction that

$$h_j = h_{j+1} + 2^{j+2} - 3.$$

Consequence

Starting from a poor assignment, the expected time can be

$$\Theta(2^n).$$


Why the basic 3-SAT result is not enough

- There are only 2^n assignments total.

Why the basic 3-SAT result is not enough

- There are only 2^n assignments total.
- A $\Theta(2^n)$-step local search is **not clearly better than brute force**.


Why the basic 3-SAT result is not enough

- There are only 2^n assignments total.
- A $\Theta(2^n)$-step local search is **not clearly better than brute force**.
- The drift away from S suggests that long runs are inefficient.

Method	Main cost	Comment
Brute force	2^n assignments	deterministic exhaustive search
Basic local search	$\Theta(2^n)$ flips	not compelling

Why the basic 3-SAT result is not enough

- There are only 2^n assignments total.
- A $\Theta(2^n)$-step local search is **not clearly better than brute force**.
- The drift away from S suggests that long runs are inefficient.

Method	Main cost	Comment
Brute force	2^n assignments	deterministic exhaustive search
Basic local search	$\Theta(2^n)$ flips	not compelling
Restarted local search	$O(n^{3/2}(4/3)^n)$ flips	faster exponential

Two key observations for improving 3-SAT

1. A uniformly random initial assignment has about $n/2$ matches with S on average.
2. With exponentially small but useful probability, it starts much closer to S.
3. Once the walk starts, the pessimistic drift points away from S.
4. Therefore: use many short random starts instead of one long run.

Two key observations for improving 3-SAT

1. A uniformly random initial assignment has about $n/2$ matches with S on average.
2. With exponentially small but useful probability, it starts much closer to S.
3. Once the walk starts, the pessimistic drift points away from S.
4. Therefore: use many short random starts instead of one long run.

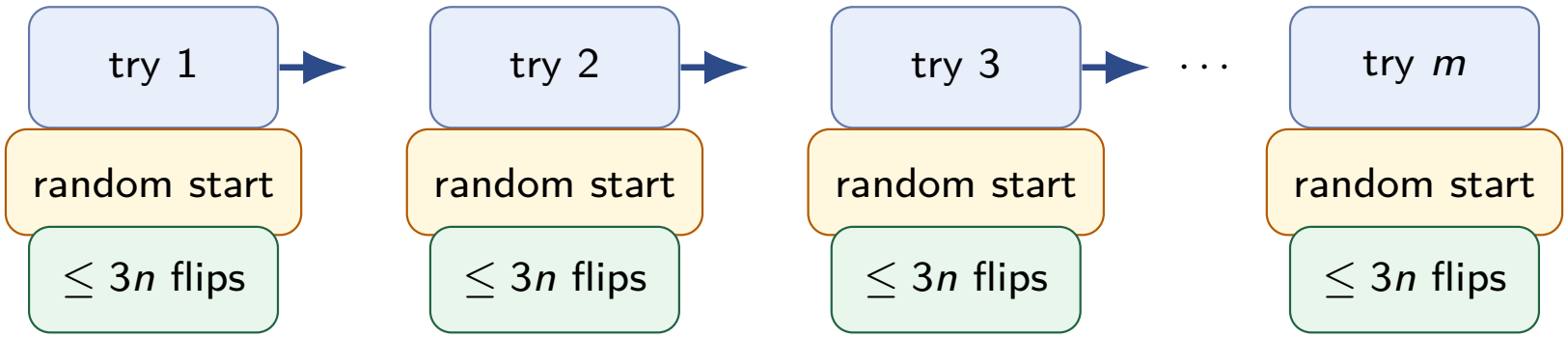
Design principle

Restart before the biased walk has too much time to drift away from the solution.


Revised Randomized 3-SAT

Modified 3-SAT Algorithm

1. Repeat up to m times:
   1. choose a random truth assignment;
   2. run at most $3n$ repair steps;
   3. stop early if a satisfying assignment is found.
2. Otherwise return "unsatisfiable."



restart rather than spend too long in a biased-down random walk

Random initial assignment

Fix a satisfying assignment S.

- Choose A_0 uniformly from $\{0,1\}^n$.
- The number of mismatches with S is

$$J \sim \mathrm{Bin}(n, 1/2).$$

- Hence

$$\Pr[J = j] = \binom{n}{j} 2^{-n}.$$


Success probability from j mismatches

Let q_j be a lower bound on the probability that one restarted trial reaches S within $3n$ flips, starting from exactly j mismatches.

- A good flip decreases the mismatch count by 1.
- A bad flip increases it by 1.

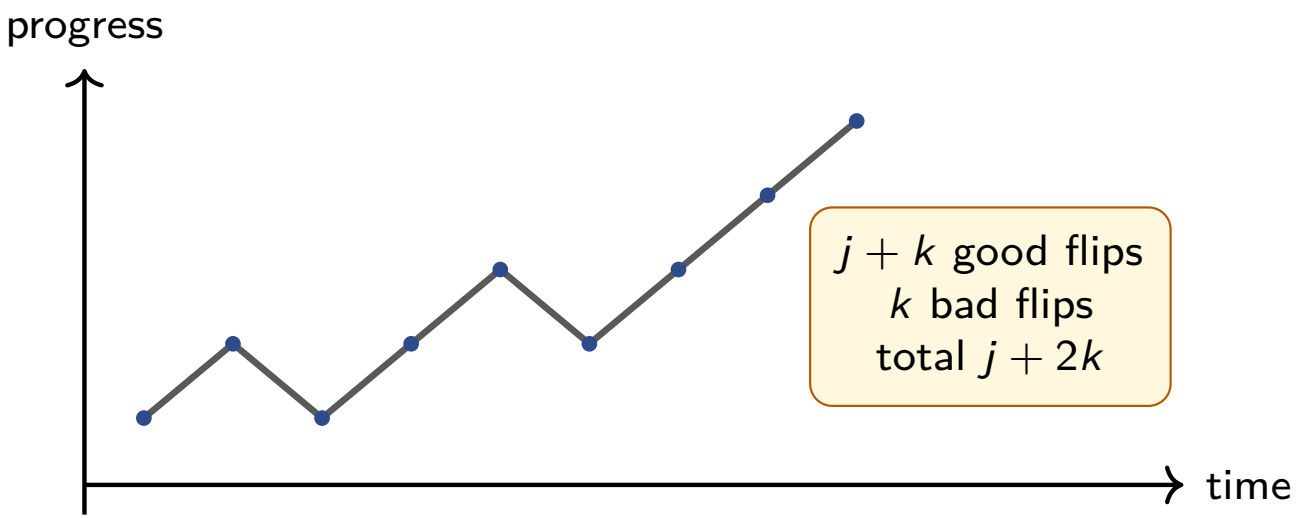
Success probability from j mismatches

Let q_j be a lower bound on the probability that one restarted trial reaches S within $3n$ flips, starting from exactly j mismatches.

- A good flip decreases the mismatch count by 1.
- A bad flip increases it by 1.
- Good flip probability is at least $1/3$.
- Bad flip probability is at most $2/3$.

We lower-bound success by counting one favorable type of path to S.


Counting a favorable path



Suppose the path uses:

- $j+k$ good flips, probability factor $(1/3)^{j+k}$;
- k bad flips, probability factor $(2/3)^k$.

The net improvement is $(j+k)-k=j$, enough to remove j mismatches.

Lower bound for q_j

If the walk has k bad flips and $j+k$ good flips, the total length is $j+2k$. The number of orders is

$$\binom{j+2k}{k}.$$

Therefore,

$$q_j \geq \max_{k=0,1,\ldots,j} \binom{j+2k}{k} \left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{j+k}.$$

- Restricting to $k \leq j$ ensures $j+2k \leq 3j \leq 3n$.


Choose the simple case $k = j$

Taking $k = j$ gives a valid path length $3j \leq 3n$:

$$q_j \geq \binom{3j}{j} \left(\frac{2}{3}\right)^j \left(\frac{1}{3}\right)^{2j}.$$

Why this is useful

The binomial coefficient counts many possible successful orders, not just one path.

Stirling's formula

Lemma (Stirling, loose form)

For $m > 0$,

$$m! = \sqrt{2\pi m}\left(\frac{m}{e}\right)^m (1 \pm o(1)).$$

In particular,

$$\sqrt{2\pi m}\left(\frac{m}{e}\right)^m \leq m! \leq 2\sqrt{2\pi m}\left(\frac{m}{e}\right)^m.$$

- We use it only to lower-bound $\binom{3j}{j}$.

Lower-bounding the binomial coefficient

For a constant $c > 0$,

$$\binom{3j}{j} = \frac{(3j)!}{j!(2j)!} \geq \frac{c}{\sqrt{j}} \left(\frac{27}{4}\right)^j.$$

Therefore,

$$\begin{aligned} q_j &\geq \frac{c}{\sqrt{j}} \left(\frac{27}{4}\right)^j \left(\frac{2}{3}\right)^j \left(\frac{1}{3}\right)^{2j} \\ &= \frac{c}{\sqrt{j}} \cdot \frac{1}{2^j}. \end{aligned}$$

- Also $q_0 = 1$.


Average over the random start

Let q be the success probability of one restarted trial.

$$q \geq \sum_{j=0}^{n} \Pr(J = j) q_j$$

$$\geq 2^{-n} + \sum_{j=1}^{n} \binom{n}{j} 2^{-n} \frac{c}{\sqrt{j} 2^j}.$$

Since $j \leq n$,

$$\frac{1}{\sqrt{j}} \geq \frac{1}{\sqrt{n}}.$$

Hence

$$q \geq \frac{c}{\sqrt{n}} \sum_{j=0}^{n} \binom{n}{j} 2^{-n} 2^{-j}.$$

The binomial theorem step

Continue from

$$q \geq \frac{c}{\sqrt{n}} \sum_{j=0}^{n} \binom{n}{j} 2^{-n} 2^{-j}.$$

Then

$$\begin{aligned} q &\geq \frac{c}{\sqrt{n}} \frac{1}{2^n} \sum_{j=0}^{n} \binom{n}{j} \left(\frac{1}{2}\right)^j (1)^{n-j} \\ &= \frac{c}{\sqrt{n}} \frac{1}{2^n} \left(1 + \frac{1}{2}\right)^n \\ &= \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n. \end{aligned}$$

Expected number of restarts & repair steps

Each restarted trial succeeds with probability at least

$$q \geq \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n.$$

The number of trials until success is geometric with mean at most

$$\frac{1}{q} = O\left(\sqrt{n} \left(\frac{4}{3}\right)^n\right).$$

Each trial runs at most $3n$ repair steps, so the expected number of repair steps is

$$O\left(n^{3/2} \left(\frac{4}{3}\right)^n\right).$$

Monte Carlo version for 3-SAT

- If the formula is unsatisfiable, the algorithm never returns a false assignment.
- If it returns an assignment, the assignment is checked and is correct.
- The only error is declaring "unsatisfiable" after not finding a solution.

Amplification idea

If the expected time to find a satisfying assignment is at most a, then using b independent blocks of length $2a$ gives failure probability at most 2^{-b}.

Monte Carlo version for 3-SAT

- If the formula is unsatisfiable, the algorithm never returns a false assignment.
- If it returns an assignment, the assignment is checked and is correct.
- The only error is declaring "unsatisfiable" after not finding a solution.

Amplification idea

If the expected time to find a satisfying assignment is at most a, then using b independent blocks of length $2a$ gives failure probability at most 2^{-b}.

- $2a \Rightarrow \frac{1}{2}$ by Markov's inequality

Monte Carlo version for 3-SAT

- If the formula is unsatisfiable, the algorithm never returns a false assignment.
- If it returns an assignment, the assignment is checked and is correct.
- The only error is declaring "unsatisfiable" after not finding a solution.

Amplification idea

If the expected time to find a satisfying assignment is at most a, then using b independent blocks of length $2a$ gives failure probability at most 2^{-b}.

- $2a \Rightarrow \frac{1}{2}$ by Markov's inequality
- independent b blocks $\Rightarrow$ error prob. $< \left(\frac{1}{2}\right)^b$.

Why restarts help

Long single run

- starts somewhere;
- biased walk tends to lose matches;
- many steps may be wasted.

Many short runs

- samples many random starting points;
- occasionally starts close to S;
- gives each promising start a short chance.

Intuitive idea

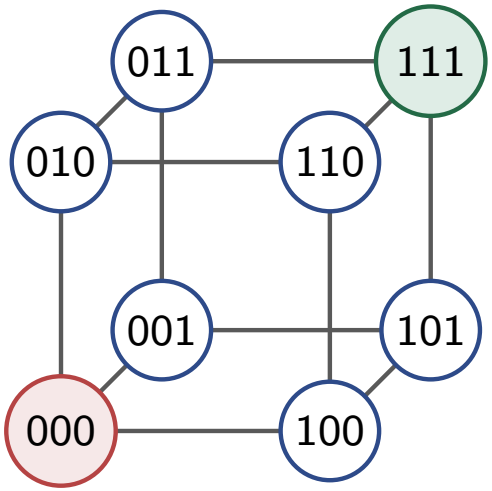
Random restarts trade one long biased walk for many independent chances to start close to S.


Comparison with brute force

Algorithm	Expected / worst-case scale	One-sided error?
brute-force	2^n assignments	*no* (always correct)
basic 3-SAT local search	$\Theta(2^n)$ steps	yes, if stopped early
restarted 3-SAT local search	$O(n^{3/2}(4/3)^n)$ steps	yes

- The base improves from 2 to 4/3.
- The algorithm is still exponential, consistent with NP-completeness of 3-SAT.


Geometric picture in assignment space



assignments form the n-dimensional hypercube

- Random restart samples a new vertex of the hypercube.
- Local search moves along hypercube edges.
- The proof tracks only the Hamming distance to one satisfying vertex S.

One-sided error in both algorithms

possible error: formula is satisfiable, but search timed out

- The algorithms never output an invalid satisfying assignment.
- Stopping early converts Las Vegas-style search into a Monte Carlo decision procedure.

Why arbitrary unsatisfied clause selection is okay

- The algorithm may choose any currently unsatisfied clause.
- The analysis does not require a random clause.
- The only property needed is:
  an unsatisfied clause under A_i is satisfied by S, so at least one of its variables disagrees with S.
- Randomness is used only in choosing which literal to flip.

Robustness

Even adversarial clause choice preserves the lower bound $1/k$ on an improving flip.


The exact process versus the comparison chain

Actual process X_i

- depends on formula structure;
- depends on current assignment;
- may not be Markovian as a scalar process.

Comparison chain Y_i

- one-dimensional;
- Markovian;
- pessimistic enough for upper bounds.

analyze Y_i to bound the algorithm

Why the cutoff $3n$ appears for restarted 3-SAT

- If the start has j mismatches, the counted favorable path uses j bad flips and $2j$ good flips.
- Total length: $3j$.
- Since $j \leq n$, this is at most $3n$.

The $3n$ cutoff is chosen to support a clean combinatorial lower bound on success probability, not because it is an exact optimal runtime.


Constants versus asymptotics

- The proof keeps constants such as $3n$ and $c/\sqrt{n}$ to make the analysis explicit.
- The main asymptotic message is the exponential base:

$$O\left(n^{3/2}\left(\frac{4}{3}\right)^n\right).$$

- Polynomial factors are less important than reducing the base from 2 to $4/3$.

An example for discussion

Consider

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3).$$

Let

$$S = (x_1, x_2, x_3) = (1, 1, 1).$$

Question

If the current assignment is $A = (0, 1, 0)$ and an unsatisfied clause is chosen, how many variables in that clause must disagree with S?

- At least one. That is all the general analysis needs.

Summary

- 2-SAT local search is controlled by an unbiased random walk on $\{0, \ldots, n\}$.
- Its expected hitting time is at most n^2.
- 3-SAT local search gives a biased walk with success direction probability only $1/3$.
- Long runs are inefficient, so random restarts are essential.
- The modified 3-SAT algorithm has expected search length

$$O\left(n^{3/2}\left(\frac{4}{3}\right)^n\right).$$

Discussion questions

1. Why is tracking matches with a fixed satisfying assignment legitimate even though the algorithm does not know S ?
2. In the 2-SAT proof, which step fails if clauses had three literals?
3. Why do restarts improve 3-SAT but are unnecessary for the polynomial 2-SAT bound?
4. What is the exact role of Markov's inequality in converting expected time into bounded-error decision algorithms?
5. Could different restart lengths improve constants or polynomial factors?

Discussions