Tutorial on Randomized Approximation Algorithms

Department of Computer Science and Engineering
Rajiv Gandhi Institute of Petroleum Technology (RGIPT), Jais

1. Motivation: When Randomness Makes Things Easier

Many optimization problems are too hard to solve exactly (NP-hard). Approximation algorithms aim to get *close to optimal* efficiently.

But what if we add randomness?

Randomized approximation algorithms use random choices during execution to achieve a good solution *in expectation*. They can be simpler, faster, and often surprisingly effective.

—

2. Classroom Warm-Up: Coin-Flipping for MAX-SAT

Scenario: You are given these 4 clauses with 3 variables:

$$(x_1 \vee \neg x_2 \vee x_3), \quad (\neg x_1 \vee \neg x_3), \quad (x_2 \vee \neg x_3), \quad (x_1 \vee x_2)$$

Each variable x_i can be True or False. Goal: Maximize number of satisfied clauses.

Activity:

1. Flip a coin for each variable. Heads = True, Tails = False.
2. Evaluate how many clauses are satisfied.
3. Repeat 5 times and note down results.

Observation: Each clause with 3 literals is unsatisfied only when all three are false. This happens with probability $(1/2)^3 = 1/8$. So each clause is satisfied with probability 7/8 on average.

Thus, the expected number of satisfied clauses:

$$\mathbb{E}[\#\text{satisfied}] = \frac{7}{8} \times (\text{total clauses})$$

Hence, the random assignment is a **7/8-approximation in expectation**.

Even pure randomness gives us a guarantee!

—

3. From Intuition to Definition

Definition: A randomized algorithm A for a maximization problem is an α-approximation in expectation if:

$$\mathbb{E}[A(I)] \geq \alpha \cdot OPT(I)$$

for all inputs I.

For minimization:

$$\mathbb{E}[A(I)] \leq \alpha \cdot OPT(I)$$

The randomness is over the algorithm's coin flips; OPT denotes the optimal solution value.

—

4. Example 1 — Randomized Algorithm for MAX-SAT

Problem: Given m clauses over n boolean variables, each clause being a disjunction (OR) of literals, find an assignment maximizing satisfied clauses.

Algorithm:

- Assign each variable independently True with probability 1/2, False otherwise.

Analysis: A clause with k literals is not satisfied only when all k literals are false:

$$P[\text{unsatisfied}] = (1/2)^k \Rightarrow P[\text{satisfied}] = 1 - (1/2)^k$$

Expected number of satisfied clauses:

$$\mathbb{E}[\text{satisfied}] = \sum_{j=1}^{m} \left(1 - (1/2)^{k_j}\right)$$

Since $OPT \leq m$,

$$\frac{\mathbb{E}[\text{satisfied}]}{OPT} \geq \min_j \left(1 - (1/2)^{k_j}\right)$$

For 3-SAT:

$$\mathbb{E}[A(I)] \geq \frac{7}{8} \cdot OPT$$

Hence, this simple algorithm is a **7/8-approximation in expectation**. We can boost confidence by repeating the random assignment several times and taking the best result (called **amplification**).

—

5. Example 2 — Randomized Approximation for MAX-CUT

Problem: Given an undirected graph $G = (V, E)$, partition V into two sets $(S, V \setminus S)$ to maximize the number of edges crossing the cut.

Algorithm:

- For each vertex $v \in V$, assign it to S with probability $1/2$.

Analysis: An edge (u, v) is cut if endpoints fall in different sets:

$$P[\text{edge cut}] = 2 \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}$$

Expected number of cut edges:

$$\mathbb{E}[\text{cut size}] = \frac{|E|}{2}$$

and

$$\mathbb{E}[\text{cut size}] \geq \frac{1}{2} OPT$$

So this is a **1/2-approximation in expectation**.

Note: Goemans & Williamson (1995) later improved this to a 0.878-approximation using semidefinite programming and randomized rounding.

—

6. Example 3 — Randomized Rounding in Set Cover

Set Cover is a classic NP-hard problem: Given universe $U = \{1, \ldots, n\}$ and subsets $S_1, S_2, \ldots, S_m$, choose the fewest subsets that cover U.

Idea: Solve the LP relaxation:

$$\min \sum c_i x_i \quad \text{s.t.} \sum_{i: e \in S_i} x_i \geq 1 \ \forall e, \quad x_i \in [0, 1]$$

Interpret x_i as probability of selecting set S_i. Then, choose each S_i independently with probability proportional to x_i. Repeat $O(\log n)$ times.

Result: The expected cost of this randomized rounding solution is $O(\log n)$ times optimal — same bound as the deterministic greedy algorithm, but the analysis is simpler and extends to weighted versions.

—

7. Why Randomization Works

- Random choices **smooth out worst cases**.
- Expected value is easier to analyze via **linearity of expectation**.
- Repetition + taking best result boosts performance.
- Randomized rounding converts fractional solutions (from LP relaxations) into feasible integer ones without losing much in expectation.

—

8. When Randomization Cannot Help: Inapproximability

For some NP-hard problems, even approximation is hard. Unless $P = NP$, there exist provable limits on achievable approximation factors.

Example: Set Cover

Greedy achieves a $\ln n$-approximation. Feige (1998) proved no algorithm can do better than $(1 - o(1)) \ln n$ unless $P = NP$. So greedy (and its randomized versions) are asymptotically optimal.

Example: Clique

Finding the largest clique cannot be approximated within factor $n^{1-\varepsilon}$ for any $\varepsilon > 0$. Even randomization or LP relaxations cannot bypass this limit.

Reason: These results rely on **gap-preserving reductions** and the **PCP theorem**, which shows that verifying NP proofs can be done by checking only a few random bits — leading directly to tight inapproximability bounds.

—

9. Design Guidelines for Randomized Approximation Algorithms

(i) Identify Probabilistic Structure

Look for natural randomness or averaging behavior:

- Independent clauses or edges (MAX-SAT, MAX-CUT)
- Fractional LP solutions that can be rounded (Set Cover, Facility Location)

(ii) Define a Simple Random Experiment

- Flip coins for binary decisions. - Sample from fractional variables. - Use random permutations or random partitions.

(iii) Analyze Expectation Using Linearity

Expected value of sum = sum of expectations — independence not required. This is the core tool for analysis.

(iv) Amplify Success Probability

Run algorithm multiple times and take the best outcome — converts expected guarantee to high probability.

(v) When to Avoid Randomization

- Problems with global constraints or dependencies (e.g., scheduling with precedence). - Problems lacking clear additive structure. - When deterministic combinatorial bounds already match known hardness limits.

—

10. Developing Insights

To design randomized approximations effectively:

- Study the structure of the optimal solution — what parts are independent?
- Check if the problem admits a "fractional relaxation" (LP, SDP).
- Experiment on small instances — does random sampling work well empirically?

- Identify symmetry — randomization works best when all variables play similar roles.

—

11. Quick FAQ Discussion

Q1: Why use randomization if deterministic greedy already works? **A:** Randomization often gives the same or better bounds with simpler analysis, and works well when deterministic heuristics are hard to reason about.

Q2: Can randomization guarantee optimality? **A:** No. It guarantees expected closeness to optimal; you can boost confidence via repetition.

Q3: Do randomized algorithms always outperform deterministic ones? **A:** Not always. For some problems (e.g., Set Cover), both have the same asymptotic ratio.

Q4: What's the role of the PCP theorem here? **A:** It defines the boundary between approximable and inapproximable problems, showing that even probabilistic checking can't beat certain ratios.

—

12. Case Studies for Further Reading

- **Goemans & Williamson (1995):** Semidefinite programming and random hyperplane rounding for MAX-CUT (0.878-approximation).
- **Feige (1998):** Tight inapproximability of Set Cover.
- **Alon & Spencer (2008):** "The Probabilistic Method" — foundational text for randomization in combinatorial optimization.
- **Vazirani (2001):** "Approximation Algorithms" — Chapter 13 covers randomized techniques comprehensively.

—

13. Summary Table

Problem	Randomized Technique	Approx. Ratio	Remarks
MAX-SAT	Random assignment	7/8	Tight bound
MAX-CUT	Random partition	1/2	0.878 via SDP
Set Cover	Randomized rounding	$O(\log n)$	Tight (Feige 1998)
Knapsack	Random sampling (rare)	$\approx (1 - \varepsilon)$	via FPTAS (deterministic)

—

14. Closing Reflection

Randomized approximation algorithms represent a bridge between combinatorial logic and probabilistic reasoning. They teach us two essential lessons:

1. Randomness can simplify complex optimization — expectation smooths out worst cases.
2. But not everything can be approximated — some barriers are fundamental, proven via reductions and PCP theory.

Mastering this perspective allows us to design new algorithms where deterministic techniques fall short — combining probability, structure, and optimization intuition.