Approximation Limits — Simple, Concrete Proofs and Examples

For B.Tech students (CS102: Introduction to Algorithms)

Department of Computer Science & Engineering

Purpose

This note shows, with simple proofs and concrete instances, two kinds of l̈imitsstudents should understand:

1. **Algorithmic tightness:** a given algorithm has a provable approximation ratio and there exist simple instances where the ratio is attained (so the analysis is tight for that algorithm).
2. **Problem-level inapproximability:** no polynomial-time algorithm can guarantee a better approximation factor (this generally requires deeper theory — we state the result and give intuition).

We cover Vertex Cover (algorithmic tightness), Max-Cut (randomized guarantee), and Set Cover (greedy bound + tight instance and comment on problem-level hardness).

1 Vertex Cover: 2-approximation (proof) and tight example

1.1 Problem

Given $G = (V, E)$, find smallest vertex set C such that every edge has at least one endpoint in C.

1.2 Algorithm (simple matching-based / greedy)

1. Initialize $C \leftarrow \emptyset$.
2. While E is non-empty:
   - Pick any edge $(u, v) \in E$.

- Add both u and v to C.
- Remove all edges incident to u or v from E.

3. Return C.

This is sometimes called the *edge-picking* algorithm.

1.3 Proof that the algorithm is a 2-approximation

Let the algorithm pick edges $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ in order. The algorithm's cover is

$$C = \{u_1, v_1, u_2, v_2, \ldots, u_k, v_k\},$$

so $|C| = 2k$.

Observe that the set of chosen edges $\{(u_i, v_i)\}_{i=1}^{k}$ is a *matching* (no two chosen edges share a vertex), because when an edge is chosen we remove all incident edges — so later choices cannot touch its endpoints.

Any vertex cover must cover every edge in this matching, and since edges in the matching are vertex-disjoint, covering k matching edges requires at least k vertices. Therefore the optimal cover size $|C^*| \geq k$.

Thus

$$|C| = 2k \leq 2|C^*|.$$

Therefore the algorithm is a 2-approximation.

1.4 Tight example (simple instance where ratio = 2)

Consider a graph consisting of k disjoint edges (i.e., k copies of K_2). Formally, $V = \{a_1, b_1, \ldots, a_k, b_k\}$ and $E = \{(a_i, b_i) : i = 1..k\}$.

- Optimal vertex cover: pick one endpoint from each edge, so $|C^*| = k$. - The algorithm may pick every chosen edge and include both endpoints, producing $|C| = 2k$.

So the algorithm achieves ratio $|C|/|C^*| = 2$ on this instance. This shows the analysis is *tight* for this algorithm.

Teaching note: This example is trivial to draw and compute — good for a quick in-class demonstration that the 2 factor can occur.

2 Max-Cut: randomized 1/2-approximation (proof) and a simple instance

2.1 Problem

Given $G = (V, E)$ undirected, partition V into S and $V \setminus S$ to maximize the number of edges crossing the cut.

2.2 Randomized algorithm

Assign each vertex independently to S with probability 1/2 (and to the other side with probability 1/2).

2.3 Analysis

Consider any edge $e = (u, v)$. The event that e is cut (its endpoints lie on different sides) occurs with probability

$$\Pr[e \text{ is cut}] = \Pr[u \in S, v \notin S] + \Pr[u \notin S, v \in S] = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.$$

By linearity of expectation, the expected number of cut edges is

$$\mathbb{E}[\text{cut size}] = \sum_{e \in E} \Pr[e \text{ is cut}] = \frac{|E|}{2}.$$

Since the maximum cut $OPT \leq |E|$, we obtain

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

Thus the randomized algorithm is a $\frac{1}{2}$-approximation in expectation.

2.4 Example

For a 4-cycle (square) with 4 edges, the random cut on average cuts 2 edges; the optimal cut can cut 4 edges; ratio $2/4 = 0.5$ — matches the guarantee.

Remark: Better approximation ratios exist (Goemans–Williamson achieves ≈ 0.878 using semidefinite programming plus randomized rounding), but that requires more advanced methods. The randomized coin-flip algorithm is excellent to teach expectation-linearity and quick reasoning.

3 Set Cover: greedy H_n-approximation (proof) and a worst-case instance

3.1 Problem

Given universe U of n elements and sets $S_1, \ldots, S_m$ (costs can be all 1 for simplicity), find minimum number of sets covering all elements.

3.2 Greedy algorithm

Repeatedly pick the set that covers the largest number of *yet-uncovered* elements. Stop when all elements are covered.

3.3 Proof of H_n (harmonic) approximation

Let OPT denote the number of sets in an optimal cover. We show greedy uses at most $OPT \cdot H_n$ sets, where

$$H_n = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n} \leq \ln n + 1.$$

Proof sketch (standard charging argument):

- Let C^* be an optimal cover with $|C^*| = OPT$.
- When the greedy algorithm begins, n elements are uncovered. At least one set in C^* must cover at least n/OPT of the uncovered elements (pigeonhole principle). So the first greedy pick covers at least n/OPT elements.
- After t elements remain uncovered, by the same argument some set in C^* covers at least t/OPT of them. Greedy picks a set covering at least that many and thus reduces the uncovered count by at least t/OPT.
- This yields a recurrence showing that the number of picks needed to reduce from t uncovered elements to $t-1$ is at most OPT/t in a charging sense.
- Summing these charges over $t = n, n-1, \ldots, 1$ yields total greedy picks $\leq OPT \cdot H_n$.

Thus greedy is an H_n-approximation.

3.4 Explicit worst-case family where greedy achieves $\Theta(\ln n)$ factor

We show a simple constructed instance where greedy performs $\approx H_n$ times worse than OPT.

Construction (standard): Let universe U be partitioned into groups with sizes decreasing like $n, n/2, n/3, \ldots, 1$ (conceptually). Create sets so that there is one "optimal" family of k sets that together cover everything (choose sets carefully), whereas greedy will pick many more sets because it always prefers sets covering the largest currently uncovered group.

A cleaner variant: the well-known "harmonic" example builds sets S_j for $j = 1, \ldots, n$ where

$$S_j = \{j, j+1, \ldots, n\}.$$

In this instance:

- The optimal cover is to take the single set S_1, so $OPT = 1$.
- The greedy algorithm will first pick S_1 (actually in this toy it picks optimal). To force greedy to behave badly, refine construction by duplicating elements and arranging sets so that at each step greedy picks a set that covers roughly n/t elements, leading to sum of reciprocals behavior.

A compact, fully rigorous worst-case construction is slightly technical to present fully here, but numerous standard references (e.g., Vazirani's textbook) give a concrete family of instances with greedy ratio arbitrarily close to H_n.

Classroom approach: For B.Tech students, present the harmonic charging proof and then show a small numerical instance (e.g., $n = 12$ with sets chosen to force greedy picks of sizes 6,4,2,...) to demonstrate greedy's suboptimality accumulating like $1+1/2+1/3+\ldots$.

3.5 Problem-level inapproximability (intuitive remark)

It is a deeper theoretical result (Feige, 1998) that no polynomial-time algorithm can guarantee an approximation factor asymptotically better than $\ln n$ for Set Cover unless $P = NP$. The proof uses advanced techniques (gap reductions, PCP). For B.Tech students, it is sufficient to state the result and point to references:

- U. Feige, "A threshold of ln n for approximating set cover", Journal of the ACM, 1998.
- Textbook: Vijay Vazirani, *Approximation Algorithms*, Chapter on Set Cover.

Thus greedy (and randomized rounding variants) achieve essentially the best possible polynomial-time approximation factor (up to constant factors).

4 What "cannot be approximated better" means (pedagogical clarification)

- **Algorithm-tightness:** We show an algorithm A has ratio α and give an input instance where A indeed returns a solution with value α times (or α worse than) optimum. This shows the *analysis* of A is tight.
- **Problem-level inapproximability:** This stronger claim says *no* polynomial-time algorithm (not only the one you analyzed) can guarantee a ratio better than some bound unless a major complexity-theoretic collapse (like $P = NP$) occurs. Proving such statements usually requires advanced results (PCP theorem, gap reductions). For B.Tech students, convey the intuition and cite results rather than proving them.

5 Teaching plan suggestions

- **Vertex Cover:** Give the 2-approx proof, draw the disjoint-edge example, ask students to run algorithm and compute ratios.
- **Max-Cut:** Present randomized algorithm, run several random trials on small graphs, compute observed average; mention Goemans–Williamson as advanced follow-up.
- **Set Cover:** Prove greedy H_n bound with charging; present a carefully chosen small instance illustrating harmonic accumulation; state Feige's hardness result informally.

6 References (for instructor reading)

- V. Vazirani, *Approximation Algorithms*, Prentice Hall, 2001.
- U. Feige, *A threshold of ln n for approximating set cover*, J. ACM, 1998.
- M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.