Randomised algorithms

April 30, 2026

1 Inclusions of complexity classes

Exercise 1.

Show that $ZPP = RP \cap co - RP$.

Solution 1.

Consider a problem in both RP and $co - RP$. Given A_1 an RP algorithm and A_2 a $co - RP$ algorithm for π, we want to design a Las Vegas algorithm for π. When A_1 accepts, the instance is guaranteed to be positive, when A_2 rejects, the instance is guaranteed to be negative. At each round, run both A_1 and A_2. If both algorithms agree, then one of them has a guaranteed answer, thus we can return the result. If they disagree, we cannot know for sure what the answer is, so we repeat this round until both algorithms agree. The probability that the A_1 and A_2 disagree is $1/2$ so in expectation, the number of rounds until both algorithms agree is two. Hence the expected running time is polynomial.

Conversely, given a Las Vegas algorithm A for a problem $\pi \in ZPP$, with expected running time $f(n)$, run A for $2f(n)$ steps. If the algorithm produces an answer before this time, return it. Otherwise, return No. If x is a negative instance, than this algorithm answers correctly all the time. If x is a positive instance, the probability that this algorithm answers no is the probability that A did not finish its execution after $2f(n)$ steps, which is by Markov inequality at most $1/2$.

Exercise 2.

Verify the following inclusions: $RP \subseteq BPP \subseteq PP$.

Exercise 3.

Show the following inclusions: $RP \subseteq NP \subseteq PP$.
Hint: for $NP \subseteq PP$, prove that 3-SAT belongs to PP.

Solution 3.

Let A be an RP algorithm. Replace all the coin flips perform by A by a non-deterministic step. We thereby obtain a non-deterministic Turing machine B. Given a negative instance x, every run of A (and so every run of B) will reject x. Given a positive instance, A accepts x with probability at least $1/2$, so some runs of A accept x, so B accepts x.

Let ϕ be a formula in CNF with three litterals per clauses, on n variables. Consider a random truth assignment. If all clauses are satisfied, accept ϕ. Otherwise, accept ϕ with probability $1/2 - 2^{-n-1}$. If ϕ is unsatisfiable, this algorithm rejects ϕ with probability $1/2 + 2^{-n-1} > 1/2$. If ϕ is satisfiable then the probability that the random truth assignment satisfies ϕ is at least 2^{-n}, so ϕ is accepted with probability $2^{-n} + (1 - 2^{-n}) \cdot (1/2 - 2^{-n-1}) = 1/2 + 2^{-2n-1} > 1/2$.

As every problem in NP admits a reduction to 3-SAT, this gives a PP algorithm for every problem in NP.

2 Finding minimal and maximal cuts

Exercise 4.

Let G be a graph. A *cut* in G is a bipartition of the vertices $A \sqcup B = V(G)$. The weight of a cut $A \sqcup B$ is the total number of edges going from A to B. The problem MAXCUT consists in finding a cut of maximum weight.

Design a randomised 1/2-approx for MAXCUT.

Solution 4.

We return a random uniform bipartition $A \sqcup B$, such random partition can be computed in linear time. Let X be the number of edges between A and B. each edge has probability $1/2$ of having its extremities in different parts of the bipartition, so $\mathbb{E}[X] = \frac{|E(G)|}{2}$ and as the maximum cut has weight at most $|E(G)|$, this gives a $1/2$ approximation algorithm.

Exercise 5. Karger's algorihtm

Given a multi-graph G, we are interested in finding a cut of minimum weight (the weight of a multi-edge between u and v is the number of edges between u and v). Let uv be an edge G. Contracting uv consists in merging the vertices u and v into one vertex w. More precisely, it removes u and v, adds a new vertex w and reattaches each edge that was incident to u or v to w (note that the common neighbours of u and v will have a multi-edge to w after this operation). We denote G/uv the graph obtained from G after contracting uv.

Karger's algorithm for the minimum cut problem constructs a sequence of graphs $G_0 = G, \dots G_{n-2}$ such that each G_i has $n - i$ vertices and is obtained from G_{i-1} by selecting an edge uv uniformly at random in G_{i-1} and setting $G_i = G_{i-1}/uv$. The graph G_n has 2 vertices a and b, the algorithm returns $A \sqcup B$ where A (resp. b) is the set of vertices that were contracted into a (resp. b).

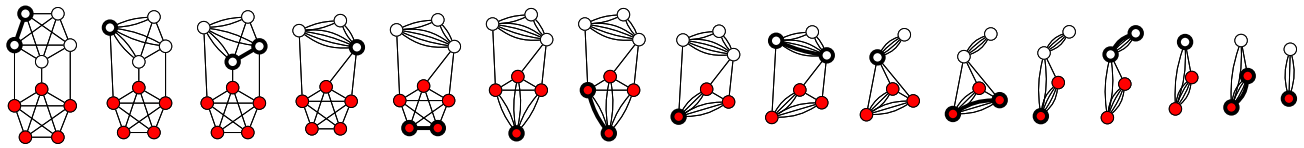

Figure 1: An execution of Karger's algorithm returning a minimum cut (credit: Thore Husfelt)

1. Argue that for every i, the weight of a minimum cut of G_i is at least that of a minimum cut of G.
2. Let $A \sqcup B$ be a cut of minimum weight in G. Show that Karger's algorithm returns $A \sqcup B$ with probability at least $\binom{n}{2}^{-1}$
3. How many times should we run Karger's algorithm to obtain a minimum cut with constant probability?
4. Suppose that instead of choosing an edge uniformly at random, one chooses a pair of adjacent vertices uniformly at random (i.e. not taking into account the number of edges between u and v) and merge them. Show that the probability of returning a minimum cut can then be exponentially small.

Solution 5.

Denote C the set of edges going from A to B and $k = |C|$.

1. Consider a cut $A'_i \sqcup B'_i$ of G_i. Let $A' \sqcup B'$ be the cut of G such that A' consists of all vertices that either belong to A'_i or were contracted into a vertex in A'_i. The weight of $A'_i \sqcup B'_i$ is equal to the weight of $A' \sqcup B'$.
2. At each step i, the probability that the algorithm selects an edge of C (conditionned on the event that none of the edges of C were previously contracted) is $\frac{k}{|E(G_i)|}$. As G_i has no cut of size less than k, the minimum degree of G_i is at least k. Hence G_i has at least $k(n-i)/2$ edges. Hence, the probability that the edge selected in the ith step does not belong to C, conditionnned on the fact that no edge of C was previously contracted, is at least $(1 - \frac{2}{n-i})$.

So the probability that the algorithm outputs $A \sqcup B$ is at least

$$\prod_{i=2}^{n}(1-\frac{2}{i}) = \prod_{i=2}^{n}\frac{i-2}{i}$$
$$= \frac{2}{n(n-1)} = \binom{n}{2}^{-1}$$

3. $O(n^2)$ times. Indeed the probability that none of n^2 independent execution of Karger's algorithm returns $A \sqcup B$ is at most $(1-\binom{n}{2})^{-1})^{n^2} \leq \exp\left(-\binom{n}{2}^{-1} \cdot n^2\right) \leq 1/e$.

4. Let K_n be a clique on n vertices. This modified version of Karger's algorithm retunrs a random uniform bipartition of the vertices of K_n. Let $A \sqcup B$ be a cut of K_n such that $2 \leq |A| \leq |B|$. The weight of this cut is $|A| \cdot |B| \geq 2 * \lceil n/2 \rceil \geq n$. On the other hand, any cut with $|A| = 1$ has weight $n-1$. So the proportion of minimum cuts is $n/2^{n-1}$ which is exponentially small.

★ Exercise 6. Goemans-Williamson Maxcut approximation scheme

A semi-definite program is a problem of the form:

$$\underset{x^{(1)},\dots x^{(n)} \in \mathbb{R}^n}{\text{maximise}} \sum_{i,j \in [n]} c_{i,j}(x^{(i)} \cdot x^{(j)}) \qquad \text{subject to } \sum_{i,j} a_{i,j,k}(x^{(i)} \cdot x^{(j)}) \leq b_k \text{ for every } k$$

where $x \cdot y = \sum_{i=1}^{n} x_i y_i$ is the scalar product between the vectors x and $y \in \mathbb{R}^n$.

We are interested in finding a Monte-Carlo approximation algorithm for the MAXCUT problem. To do so, we follow an approach similar to algorithm described for MAXSAT in the lecture: first express the problem as an integer integer quadratic program (instead of integer linear program for MAXSAT), then relax this integer quadratic program (IQP) into a semi-definite program (SDP) and solve it, finally, round randomly the solution of this SDP to approximate the solution of the IQP.

1. Let G be a graph on n vertices, let $A \sqcup B$ be a cut of G. For each vertex u, let $x^{(u)} = 1$ if $u \in A$ and $x^{(u)} = -1$ if $u \in B$. Express the weight of the cut $A \sqcup B$ as a polynomial in the variables $(x^{(u)})_{u \in V(G)}$. Let $f((x^{(u)})_{u \in V(G)})$ be the corresponding objective function.

2. For each vertex u, let $y^{(u)} = (1, 0, \dots 0) \in \mathbb{R}^n$ if $u \in A$ and $y^{(u)} = (-1, 0 \dots 0) \in \mathbb{R}^n$ if $u \in B$. Express the weight of the cut $A \sqcup B$ as a polynomial in the variables $(y^{(u)} \cdot y^{(v)})_{u,v \in V(G)}$. Let $g((y^{(u)})_{u \in V(G)})$ be the corresponding objective function.

3. We now relax the constraint $y^{(u)} = (\pm 1, 0, \dots 0)$ and replace it by $\left\|y^{(u)}\right\|_2 = 1$. Argue that the corresponding optimisation problem is a semi-definite program.

4. Denote $(\hat{y}^{(u)})_{u \in V(G)}$ a solution of this semi-definite program. Informally, when is some edge uv contributing to the objective function?

5. We now do the randomised rounding step. Define a way of sampling random variables $X^{(u)} \in \{-1, +1\}$ for $u \in V(G)$ such that $\mathbb{E}[f((X^{(u)})_{u \in V(G)})] \geq 0.878 \cdot g((\hat{y}^{(u)})_{u \in V(G)})$.

6. Conclude.

★ Solution 6.

1. The weight of the cut is $f((x^{(u)})_{u \in V(G)}) = \sum_{uv \in E(G)} \frac{1 - x^{(u)}x^{(v)}}{2}$.

2. The weight of the cut is $g((y^{(u)})_{u \in V(G)}) = \sum_{uv \in E(G)} \frac{1 - (y^{(u)} \cdot y^{(v)})}{2}$.

3. We have $g((y^{(u)})_{u \in V(G)}) = h((y^{(u)})_{u \in V(G)}) + |E(G)|/2$, where $h((y^{(u)})_{u \in V(G)}) = \sum_{uv \in E(G)} -\frac{1}{2}(y^{(u)} \cdot y^{(v)})$. Hence, optimising f is the same as optimising h which we will now do without loss of generality. The constraint $\left\|y^{(u)}\right\|_2$ can be encoded by $(y^{(u)} \cdot y^{(u)}) \leq 1$ and $-(y^{(u)} \cdot y^{(u)}) \leq -1$.

4. The edge uv contributes the most when the scalar product between $\hat{y}^{(u)}$ and $\hat{y}^{(v)}$ is close to -1, i.e. when $\hat{y}^{(u)}$ and $\hat{y}^{(v)}$ are close to being antipodal on the sphere.

5. Sample a random vector z of norm 1. Let $X^{(u)} = 1$ if $z \cdot \hat{y}^{(u)} \geq 0$ and $X^{(u)} = -1$ if $z \cdot \hat{y}^{(u)} < 0$. If $\hat{y}^{(u)}$ and $\hat{y}^{(v)}$ are close to being antipodal, then with high probability, $X^{(u)}$ and $X^{(v)}$ will have different signs.

   The probability that $X^{(u)}$ and $X^{(v)}$ have different signs is proportional to the angle θ_{uv} between $\hat{y}^{(u)}$ and $\hat{y}^{(v)}$:

   $$\mathbb{P}(\hat{y}^{(u)} \text{ and } \hat{y}^{(v)} \text{ are on different sides of the hyperplane normal to } z) = \frac{\theta_{uv}}{\pi}$$

   By linearity of expectation, we get $\mathbb{E}[f((X^{(u)})_{u \in V(G)})] = \sum_{uv \in E(G)} \frac{\theta_{uv}}{2}$. On the other hand, $g((\hat{y}^{(u)})_{u \in V(G)}) = \sum_{uv \in E(G)} \frac{1-\cos(\theta_u v)}{2}$. As $\frac{2\theta}{\pi(1-cos(\theta))} \geq \alpha \approx 0.878$, we get that $\mathbb{E}[f((X^{(u)})_{u \in V(G)})] \geq \alpha g((\hat{y}^{(u)})_{u \in V(G)})$ which is at least α times the weight of the maximum cut of G.