COS 521: Advanced Algorithms
Final

Zachary Hervieux-Moore

Friday 20$^{\text{th}}$ January, 2017

I pledge my homor that I have not violated the honor code during this examination.

Signature: *Zachary Hervieux-Moore*

Exercise 1: In this problem you will explore a simple search algorithm for 2SAT. Given a 2SAT formula with m constraints and n variables, do the following:

Start with z_0 as uniformly random assignment to the n Boolean variables. In rounds $r = 0, 1, 2, \ldots, N$, do:

a) If z_i satisfies all constraints, return z_i.

b) Otherwise, choose a random constraint C that is not satisfied by z_i.

c) Choose one of the two variables at random from C and flip the value of this variable in assignment z_i to obtain z_{i+1}.

Show that given a satisfiable 2SAT formula, this algorithm succeeds with high probability in finding a satisfying assignment when $N = n^c$ for some constant c.

Answer: First, let us denote by S the set of all the configurations that satisfy the given formula. Now, let S_i be the satisfiable configuaration that differs from the z_i configuration by the least. We now denote this difference by d_i. We now show that d_i is a random walk. Suppose $d_i > 0$, then z_i does not satisfy all constraints. Then, we pick a constraint C at random. There are 2 possibilities. If both variables in clause C are different than their respective variables in S_i, then d_i will be guaranteed to decrease by 1. If one of them is correct and the other is wrong, then, with probability $1/2$, d_i will increase by 1 or, with probability $1/2$, d_i decreases by 1. Putting this together, we have the transition probabilities

$$p(d_{i+1} = k + 1 | d_i = k) \leq 1/2$$
$$p(d_{i+1} = k - 1 | d_i = k) \geq 1/2$$

That is, in the worst case, d_i behaves like a random walk. We now wish to calculate the time it takes for d_i to reach 0. In the worst case, the walk starts from n. From Lecture 10, example 1, i will need to be on the order of n^2 for there to be a good chance of d_i to reach 0. We conclude by saying that $N = O(n^2)$ will make the algorithm succeed with high probability.

Exercise 2: Give a poly(n) time algorithm to solve the following linear program for an arbitrary cost vector $c \in \mathbb{R}^{2n}$:

$$\begin{aligned} &\min c^T x \\ \text{s.t. } &\sum_{i \in S} x_i \geq 1 \quad \forall S \subseteq [2n], |S| = n \\ &x_i \geq 0 \quad \forall 1 \leq i \leq 2n \\ &x \in \mathbb{R}^{2n} \end{aligned}$$

Answer: First let us note that there is $\binom{2n}{n}$ constraints since $|S| = n$ and $S \subseteq [2n]$. We also note that the rows of A will sum up to n when we write the LP as

$$\begin{aligned} &\min c^T x \\ \text{s.t. } &Ax \geq \mathbf{1} \\ &x \geq 0 \end{aligned}$$

Where $\mathbf{1}$ is the all ones vector. We now write the dual,

$$\begin{aligned} &\max \mathbf{1}^T y \\ \text{s.t. } &A^T y \leq c \\ &y \geq 0 \end{aligned}$$

Where the objective simplifies to

$$\begin{aligned} &\max \sum_{i=1}^{\binom{2n}{n}} y_i \\ \text{s.t. } &A^T y \leq c \\ &y \geq 0 \end{aligned}$$

Using the fact that the rows of A sum up to n, the dual has the following implication if we sum up all the constraints

$$\begin{aligned} A^T y \leq c \implies n \cdot \sum_{i=1}^{\binom{2n}{n}} y_i \leq \sum_{i=1}^{2n} c_i \\ \iff \sum_{i=1}^{\binom{2n}{n}} y_i \leq \frac{1}{n} \sum_{i=1}^{2n} c_i \end{aligned}$$

Notice that this is an upper bound on the objective. By the strong duality theorem, if we can achieve this, then the solution is optimal for the primal problem. First, suppose that $x_i > 0$, then by complementary slackness, the dual constraints are all binding and the inequality above becomes an equality. Then, setting $x_i = 1/n$ will yield the optimal value of $\frac{1}{n}\sum_{i=1}^{2n} c_i$.

Now, suppose there are j of the $x_i = 0$. Then, we can remove the i^{th} columns of A and c to get A' and c'. We get that the new dual problem implies

$$A'^T y \leq c' \implies (n-j) \cdot \sum_{i=1}^{\binom{2n}{n}} y_i \leq \sum_{i=1}^{2n-j} c_i$$
$$\iff \sum_{i=1}^{\binom{2n}{n}} y_i \leq \frac{1}{n-j} \sum_{i=1}^{2n-j} c_i$$

Again, by setting $x_i = 1/(n-j)$, this yields the optimal value. Note that $j < n$ since if $j = n$, this violates the constraints of the primal. It should also be noted, that if some of the $x_i = 0$, the ones that would equal 0 are the ones with the greatest corresponding c_i values. This is evident from the primal problem. Thus, we now have the ground work for the algorithm.

First, ensure that c has the form $c_1 \leq c_2 \leq \ldots \leq c_{2n}$. Then compute the following

$$\begin{aligned}
v_0 &= c^T x, \text{ where } x = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}) \\
v_1 &= c^T x, \text{ where } x = (\frac{1}{n-1}, \frac{1}{n-1}, \ldots, \frac{1}{n-1}, 0) \\
&\vdots \\
v_{n-j} &= c^T x, \text{ where } x = (\frac{1}{n-j}, \frac{1}{n-j}, \ldots, \frac{1}{n-j}, \underbrace{0, \ldots, 0}_{j \text{ times}})
\end{aligned}$$

Then choose the smallest v_i and pick the corresponding x value. This algorithm performs $2n$ multiplications $n-j$ times, therefore the runtime will be $O(n^2)$.

Exercise 3: Can a group of m computers even agree about the time of the day, if some q of them are faulty (perhaps maliciously so)? We formalize a very simple sub-case where they have to agree on a single bit.

Initially, each computer holds a bit $b_i \in \{0, 1\}$. They need to communicate with each other so as to agree on a bit b_{agree} at the end of the communication. No computer knows which of the others are good. So the desired outcome is as follows. If all the initial bits b_i for the good computers are equal, then b_{agree} must equal this b_i for any good computer. Otherwise, b_{agree} is allowed to be either 0 or 1, provided that all good computers set b_i to the same b_{agree}.

This question will assume $m = 3q+1$. We also assume that all computers have access to a *global random coin* at each step, which is a random bit that is made available to all of them, and is renewed at each step.

Show that the following protocol converges in $O(1)$ expected steps, no matter what the q malicious computers do.

Each computer maitains an *Opinion* bit - at the start of the i^{th} computer sets this to b_i. Communication proceeds in rounds and in each round, every computer sends its *Opinion* bit to every other computer. Each computer considers all the m values of the *Opinion* bits and computes a *Popular* bit - which is the majority of the *opinion* bits - and *Multiplicity*, the number of times the popular bit appeared in the m *opinion* bits.

1) If *Multiplicity* $\geq 2q + 1$, set *Opinion* = *Popular*.

2) If *Multiplicity* $\leq 2q$, set *Opinion* to the global random bit.

Notice that the faulty/malicious computers may not follow the protocol above and in particular, could send different bits to different computers in the same round.

1) Show that if in some round all the opinion bits of the good computers are equal, then no opinion bit of a good computer changes thereafter.

2) Show that with 99 percent probability, the good computers agree in their opinion bits in $O(1)$ number of rounds. (Hint: Show that the probability of good computers not agreeing on their opinion bits after the t^{th} round is at most 2^{-t+1}).

3) Modify the protocol above so that the good computers can detect *when* an agreement among all good computers has been reached.

Answer:

1) If all the good computers have equal opinion bits, that means that $m - q = 2q + 1$ good computers have the same opinion bits. Thus the multiplicity of the popular bit will always be at least $2q + 1$, which satisfies the condition for the *Opinion = Popular*. Thus, the good computers won't change their bits and this will continue in perpetuity.

2) Since $m = 3q + 1$ and there are q bad computers, that means there are $2q + 1$ good computers. Thus, there is always an odd amount. Which means, at initialization, that there will be a group of good computers with $b_i = 1$ and another group with $b_i = 0$. Wlog, assume the group with $b_i = 1$ is larger. The number of computer in this group will be greater than $q+1$ since $\lceil \frac{2q+1}{2} \rceil = q+1$. Again, with no loss in generality, assume this group has $q + 1$ computers.

   Now, there are two possible. Either all the malicious send bits equal to 1. In which case, these computers will receive $2q + 1$ of them and will hence set $b_i = 1$. Other wise, they set b_i to the random coin. This is true for both the large and small group of good computers. Notice that the malicious computers will not send 1's to every computer as this will end the game. They will make some computers set the popular bit to 1 or force them to take the coin.

   Let's go through the coin flips. If the coin produces a 1, then all the computers will be in agreement as the popular bit is a 1. If the coin is a 0, then all the computers that don't receive all 1's will set $b_i = 0$. The malicious computers can choose any number of them to do this. The only thing that matters is that there is a larger group and a smaller group. Again, the game continues with the same outcomes except for the larger group may have $b_i = 0$. The only thing this changes is the flip of the coin that ends the game. However, both flips happen with the same probability. Thus, there is a $1/2$ chance that all computers will agree after every round. That is, the probability that the computers agree after time t is

$$P(\text{computers agree after round } t) = 1 - 2^{-t+1}$$

   Where the $+1$ is due to the first round being a wash since it is initialization. Thus is takes roughly 8 turns for there to be 99% chance that they do not agree.

3) The modification of the protocol requires keeping track of the global coin flip. That is, if you're a good computer, keep track of the coin flip. There is only one condition required for you to know if you are all in agreement. It is that the coin flip must be the same as the popularity bit. This forces the smaller group to become the larger group. This is enough to ensure that all the good computers are in agreement.

Exercise 4: In class, we designed an algorithm for the MAX-SAT problem using a Linear Programming relaxation and a simple rounding scheme. Here, we explore a similar algorithm for the MAX-AND Problem: given a DNF (Disjunctive Normal Form) formula (where each clause/constraint is an AND of an arbitrary number of literals), find an assignment that satisfies the maximum possible number of clauses.

a) Given an instance I of the MAX-AND problem with m clauses and n constraints, write a linear programming relaxation with poly(m, n) constraints and describe a simple randomized rounding scheme with the following guarantee: if the best assignment for the instance I satisfies $(1-\epsilon)m$ clauses, your rounding should output a solution that satisfies at least $(1-2\epsilon)m$ clauses.

b) Can you make your rounding scheme above deterministic?

c) EXTRA CREDIT: Suppose each clause in the MAX-AND instance has at most k literals. Can you give an improved algorithm that guarantees that the output assignment satisfies $(1-2(1-1/2^k)\epsilon)m$ clauses given that the best assignment satisfied $(1-\epsilon)m$ clauses in the instance?

Answer:

a) We first notice that maximizing a DNF is the dual to minimizing CNF by the simple negation of the expression and by de Morgan's law. Thus, we now have the prototypical min SAT problem

$$\begin{aligned} \min & \sum_j z_j \\ \text{s.t.} & \sum_{i=1}^{k} y_{j_i} \leq z_j \quad \forall j \\ & 0 \leq z_j \leq 1 \quad \forall j \\ & 0 \leq y_i \leq 1 \quad \forall i \end{aligned}$$

Which we compare to the min vertex cover problem.

$$\begin{aligned} \min & \sum_i w_i x_i \\ & 0 \leq x_i \leq 1 \quad \forall i \\ x_i + x_j \geq & \quad \forall \{i,j\} \in E \end{aligned}$$

These two problems are very similar. In fact, one can see that solving the min vertex cover solves the min SAT since $1 \geq z_j$ and we restrict the sums to all pairwise combinations in the clauses. Now, we have that the optimal min cost is $1 - (1 - \epsilon) = \epsilon$. I am not able to get an approximation of 2 (which you'd expect from min vertex cover), but I'll show my best bound. Round by letting $y_i = 1$ with probability y_i and 0 otherwise. Then

$$\begin{aligned} P(\text{clause } j \text{ not satisfied}) &= \prod_i (1 - y_{j_i}) \\ &\leq \left(\frac{\sum_i (1 - y_{j_i})}{k} \right)^k \\ &= \left(\frac{1 - \sum_i (y_{j_i})}{k} \right)^k \\ &\leq \left(\frac{1 - z_j}{k} \right)^k \\ &\leq e \end{aligned}$$

Where the penultimate inequality is satisfied due to the LP constraints and the last inequality is a rudimentary calculus result. Thus, we have the following guarantee on the original problem: $(1 - e \cdot \epsilon)m$ clauses will be satisfied.

b) We can use the deterministic rounding algorithm presented in class that says to set y_i to 0 is $y_i < 1/2$ and set y_i to 1 otherwise. We showed that this has at most 2OPT. In this cause, OPT is ϵ so 2OPT is 2ϵ as before and we get that we can satisfy $(1 - 2\epsilon)$ clauses.

Exercise 5: In class, we studied the continuous Lovasz extension of a submodular function $f : \{0,1\}^n \to \mathbb{R}$ (or equivalently, $f : 2^{[n]} \to \mathbb{R}$) - $F : [0,1]^n \to \mathbb{R}$ defined by $F(x) = \mathbb{E}_\lambda[f(\{i|x_i \geq \lambda\})]$ where λ is drawn uniformly from $[0,1]$.

We now study a different extension and relate it to the Lovasz extension. Define

$$G(x) = \mathbb{E}_{y \sim D_x}[f(y)]$$

where $y \sim D_x$ is sampled by choosing y_i to be $+1$ with probability x_i and 0 otherwise independently for each i.

Prove that $G(x) \geq F(x)$ for every x.

Hint: For $S_i = \{1, 2, \ldots, i\}$ for every $0 \leq i \leq n$ and any set $X \subseteq [n]$, observe that for any (not necessarily submodular) function $f : 2^{[n]} \to \mathbb{R}$, $f(X) = f(\emptyset) + \sum_{i=1}^{n} f(X \cap S_i) - f(X \cap S_{i-1})$.

Answer: We note the simplification of the Lovasz extension shown in class

$$F(x) = (1 - x_1)f(\emptyset) + \sum_{i=1}^{n-1}(x_i - x_{i+1})f(S_i) + x_n f(S_n)$$

where $1 \geq x_1 \geq x_2 \geq \ldots \geq x_n \geq 0$. We now show the equality holds for $n = 1, 2$ then try to show it in general. For $n = 1$, we have

$$F(x) = (1 - x_1)f(\emptyset) + x_1 f(S_1)$$
$$G(x) = \mathbb{E}_{y \sim D_x}[f(y)] = \sum_{y \in 2^{[1]}} f(y)p(y) = f(S_1)x_1 + (1 - x_1)f(\emptyset)$$

Thus, the two are equal. Now for $n = 2$. The Lovasz extension is

$$F(x) = (1 - x_1)f(\emptyset) + (x_1 - x_2)f(S_1) + x_2 f(S_2)$$

By the definition of $G(x)$ we have

$$G(x) = x_1 x_2 f(S_2) + x_1(1 - x_2)f(S_1) + (1 - x_1)x_2 f(\{2\}) + (1 - x_1)(1 - x_2)f(\emptyset)$$

Now we add and subtract a term so that we can use submodularity

$$G(x) = x_1 x_2 f(S_2) + x_1(1 - x_2)f(S_1) + (1 - x_1)x_2 f(\{2\}) + (1 - x_1)x_2 f(S_1) - (1 - x_1)x_2 f(S_1) + (1 - x_1)(1 - x_2)f(\emptyset)$$

Now we note that $S_1 \cap \{2\} = \emptyset$ and $S_1 \cup \{2\} = S_2$ and use the definition of submodularity that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.

$$\begin{aligned} G(x) &\geq x_1 x_2 f(S_2) + x_1(1-x_2) f(S_1) + (1-x_1) x_2 f(S_2) \\ &\quad + (1-x_1) x_2 f(\emptyset) - (1-x_1) x_2 f(S_1) + (1-x_1)(1-x_2) f(\emptyset) \\ &= x_2 f(S_2) + (x_1 - x_2) f(S_1) + (1-x_1) f(\emptyset) \\ &= F(x) \end{aligned}$$

Thus, for $n = 2$, we have $G(x) \geq F(x)$. Hopefully this elucidates some of the hidden forces at work in the general case which we now show. Using the hint we have

$$\mathbb{E}_{y \sim D_x}[f(y)] = \sum_{y \in 2^{[n]}} f(y) p(y)$$

$$= \sum_{y \in 2^{[n]}} \left[f(\emptyset) + \sum_{i=1}^{n} f(y \cap S_i) - f(y \cap S_{i-1}) \right] p(y)$$

$$= f(\emptyset) + \sum_{y \in 2^{[n]}} \left[\sum_{i=1}^{n} f(y \cap S_i) - f(y \cap S_{i-1}) \right] p(y)$$

Now using $f(S) - f(S \cap T) \geq f(S \cup T) - f(T)$ again with $S = y \cap S_i$ and $T = S_{i-1}$

$$\geq f(\emptyset) + \sum_{y \in 2^{[n]}} \left[\sum_{i=1}^{n} f(y \cap S_i \cup S_{i-1}) - f(S_{i-1}) \right] p(y)$$

Note that $y \cap S_i \cup S_{i-1}$ is either S_{i-1} or S_i. If it is S_{i-1}, then the sum term is 0. Otherwise, $i \in y$. Thus, we have

$$= f(\emptyset) + \left[\sum_{i=1}^{n} f(S_i) - f(S_{i-1}) \right] p(y \cap S_i \cup = S_i)$$

$$= f(\emptyset) + \left[\sum_{i=1}^{n} f(S_i) - f(S_{i-1}) \right] p(y_i = 1)$$

$$= f(\emptyset) + \left[\sum_{i=1}^{n} f(S_i) - f(S_{i-1}) \right] x_i$$

Rewriting this sum,

$$= (1 - x_1)f(\emptyset) + \sum_{i=1}^{n-1}(x_i - x_{i+1})f(S_i) + x_n f(S_n)$$
$$= F(x)$$

Thus, we have shown that $G(x) \geq F(x)$.

Exercise 6: Let $\mu, \nu \in \mathbb{R}^n$ be a pair of orthogonal unit vectors and $\lambda \in [0, 1]$. Let $X \in \mathbb{R}^n$ be a random variable sampled as follows. With probability λ sample each coordinate x_i of x independently from $\mathcal{N}(\mu_i, 1)$ and with probability $1 - \lambda$, sample each coordinate x_i of x independently from $\sqsubseteq_\rangle, \infty$. Here $\mathcal{N}(m, \sigma)$ is the gaussian distribution with mean m and standard deviation σ studied in class with the probability density function $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}}$. This problem explores an algorithm to estimate unknown μ, ν, and λ from samples of the random variable X.

1) Consider the third moment tensor $T \in \mathbb{R}^{n \times n \times n}$ for the random variable x, $T_{i,j,k} = \mathbb{E}[x_i x_j x_k] - \sum_{i=1}^n E[x] \otimes e_i \otimes e_i - \sum_{i=1}^n e_i \otimes E[x] \otimes e_i - \sum_{i=1}^n e_i \otimes e_i \otimes E[x]$. Show that T has rank 2.

2) Estimate T using samples from x. How many samples are required to estimate each entry of $T_{i,j,k}$ to within an additive error of ϵ correctly with probability at least $1 - 1/n^10$ (over the samples)?

3) Give an efficient algorithm to computer $\tilde{\lambda}, \tilde{\mu}, \tilde{\nu}$ so that $\|\mu - \tilde{\mu}\|_2 < 1/10$, $\|\nu - \tilde{\nu}\|_2 < 1/10$, and $\|\lambda - \tilde{\lambda}\|_2 < 1/10$. You may use without proof, the guarantees for Jennrich's algorithm for tensor decomposition discussed in class.

Answer:

1) First, for simplicity, let's denote $\mathbb{E}[x] = \lambda\mu + (1 - \lambda)\nu = \bar{x}$. Then, the tensors being subtracted are

$$(\sum_{l=1}^n \bar{x} \otimes e_l \otimes e_l)_{i,j,k} = \begin{cases} \bar{x}_i, & \text{for } j = k \\ 0, & \text{otherwise} \end{cases}$$

$$(\sum_{l=1}^n e_l \otimes \bar{x} \otimes e_l)_{i,j,k} = \begin{cases} \bar{x}_j, & \text{for } i = k \\ 0, & \text{otherwise} \end{cases}$$

$$(\sum_{l=1}^n e_l \otimes e_l \otimes \bar{x})_{i,j,k} = \begin{cases} \bar{x}_k, & \text{for } i = j \\ 0, & \text{otherwise} \end{cases}$$

Also, the first component is equal to

$$\mathbb{E}[x \otimes x \otimes x]_{i,j,k} = \begin{cases} \mathbb{E}[x_i]\mathbb{E}[x_j]\mathbb{E}[x_k] & \text{for } i \neq j \neq k \\ \mathbb{E}[x_i^2]\mathbb{E}[x_k] & \text{for } i = j \neq k \\ \mathbb{E}[x_i^2]\mathbb{E}[x_j] & \text{for } i = k \neq j \\ \mathbb{E}[x_j^2]\mathbb{E}[x_i] & \text{for } j = k \neq i \\ \mathbb{E}[x_i^3] & \text{for } i = j = k \end{cases}$$

Where the components being independent allows us to break the expectation products. Now, note that the second and third moments of a Gaussian mixture are just the weighted sums of the second and third moments of the individual components. This is verified by iterated expectation and conditioning on the mixture.

$$\mathbb{E}[x_i^2] = 1 + (\lambda\mu^2 + (1-\lambda)\nu^2)$$
$$\mathbb{E}[x_i^3] = \lambda\mu^3 + \lambda 3\mu + (1-\lambda)\nu^3 + (1-\lambda)3\nu$$

Putting both results together, we get

$$\mathbb{E}[x \otimes x \otimes x] - \sum_{i=1}^{n} E[x] \otimes e_i \otimes e_i - \sum_{i=1}^{n} e_i \otimes E[x] \otimes e_i - \sum_{i=1}^{n} e_i \otimes e_i \otimes E[x]$$

$$= \begin{cases} \bar{x}_i\bar{x}_j\bar{x}_k & \text{for } i \neq j \neq k \\ [1 + (\lambda\mu_i^2 + (1-\lambda)\nu_i^2)]\bar{x}_k - \bar{x}_k & \text{for } i = j \neq k \\ [1 + (\lambda\mu_i^2 + (1-\lambda)\nu_i^2)]\bar{x}_j - \bar{x}_j & \text{for } i = k \neq j \\ [1 + (\lambda\mu_j^2 + (1-\lambda)\nu_j^2)]\bar{x}_i - \bar{x}_i & \text{for } j = k \neq i \\ \lambda\mu_i^3 + (1-\lambda)\nu_i^3 & \text{for } i = k = j \end{cases}$$

$$= \begin{cases} [\lambda\mu_i + (1-\lambda)\nu_i][\lambda\mu_j + (1-\lambda)\nu_j][\lambda\mu_k + (1-\lambda)\nu_k] & \text{for } i \neq j \neq k \\ [\lambda\mu_i^2 + (1-\lambda)\nu_i^2][\lambda\mu_k + (1-\lambda)\nu_k] & \text{for } i = j \neq k \\ [\lambda\mu_i^2 + (1-\lambda)\nu_i^2][\lambda\mu_j + (1-\lambda)\nu_j] & \text{for } i = k \neq j \\ [\lambda\mu_j^2 + (1-\lambda)\nu_j^2][\lambda\mu_i + (1-\lambda)\nu_i] & \text{for } j = k \neq i \\ \lambda\mu_i^3 + (1-\lambda)\nu_i^3 & \text{for } i = k = j \end{cases}$$

From this, it is evident what the 2 rank decomposition is

$$= \lambda(\mu \otimes \mu \otimes \mu) + (1-\lambda)(\nu \otimes \nu \otimes \nu)$$

2) We now need to estimate the values in $T_{i,j,k}$ which all depend on the means of the mixture μ and ν. Thus, if we can get the mean $\bar{x}$ close, all the other values will get close up to some linear scale. Let $\widehat{x}$ denote the estimate of the mean $(1/n \sum_i x_i)$. Using Hoeffding's inequality,

$$P(\widehat{x} - \mathbb{E}[\widehat{x}] \leq \epsilon) \leq 1 - e^{-2s\epsilon}$$

Where s is the number of samples. Thus, we want $e^{-2s\epsilon} \leq 1/n^{10}$. So we choose $s = O(\frac{5 \log n}{\epsilon})$ to ensure that we get the accuracy we want with the probability specified.

3) We simply use Jennrich's algorithm to extract $\lambda(\mu \otimes \mu \otimes \mu)$ and $(1 - \lambda)(\nu \otimes \nu \otimes \nu)$ since Jennrich's algorithm extracts the decomposition. This algorithm is polynomial in n and $1/\epsilon$. We then apply the tensor power iteration method proposed in the last problem of homework 4 to recover $(\mu \otimes \mu \otimes \mu)$ and $(\nu \otimes \nu \otimes \nu)$ which is efficient due to quadratic convergence. We then extract λ by checking the scalar difference between these two answers.