Section 1 chapter 1 : Intro to approximation algorithms

1. **α-approximation algorithms** (minimization: $opt \leq f \leq \alpha\, opt$ for $\alpha > 1$)

2. **Polynomial-time approximation scheme** (PTAS) is a family of algorithm $\{A_\epsilon\}$ to a problem with $(1+\epsilon)$-approximation algorithm (for minimization) and $(1-\epsilon)$-approximation algorithm (for maximization)

3. **Set Cover** Given $E = \{e_1, \cdots, e_n\}$, subsets $S_1, \cdots, S_m \subseteq E$ Find $I \subseteq \{1, \cdots, m\}$ such that $\sum_{j \in I} w_j$ minimized while $\cup_{j \in I} S_j = E$, i.e. set of S_j covers E

4. **Weighted Vertex Cover** as a specialized case for set cover. Given $G = (V, E)$, and $w_v \geq 0$ for each $v \in V$, goal is to find $C \subseteq V$ such that for all $(u, v) \in E$, $u \in C$ or $v \in C$. We can convert vertex cover to a set cover problem by noticing that E is the set we want to cover and let S_v of weight w_v be edges incident to vertex $v \in V$. Note for any vertex cover C there is a set cover I of same weight.

5. **Unweighted Vertex Cover** $w_v = 1$ for all $v \in V$

6. **LP formulation and Relaxation** Let x_v be decision variables that represent the decision that $S_v = \{e \in E : v \in e\}$ is included in the solution, i.e. $x_v = 1$ implies $v \in C$

$$\begin{aligned} &\min && \sum_{j=1}^{n} w_j x_j \\ &s.t. && \\ &\forall e \in E && \sum_{j: e \in S_j} x_j \geq 1 \\ &\forall v \in V && x_v \in \{0,1\} \end{aligned} \quad \xrightarrow{\text{relaxation}} \quad \begin{aligned} &\min && \sum_{j=1}^{n} w_j x_j \\ &s.t. && \\ &\forall e \in E && \sum_{j: e \in S_j} x_j \geq 1 \\ &\forall v \in V && x_v \geq 0 \end{aligned}$$

Note every feasible solution for IP is feasible for LP. Let Z_{IP}^* and Z_{LP}^* be optimal value for integer and the relaxed linear program, and OPT be optimal value of the problem, then

$$Z_{LP}^* \leq Z_{IP}^* = OPT$$

for minimization problem

7. **Deterministic Rounding** Given LP solution x^*, include S_v in solution if and only if $x_v^* \geq {}^1\!/_f$ where $f_e = |\{v : e \in S_v\}|$ represent number of times e is included in some S_v and $f = \max_{e \in E} f_e$ represents maximum number of times any e appears in S. Equivalent to rounding to get an approximate integer solution

$$\hat{x}_v = \begin{cases} 1 & x_v^* \geq \dfrac{1}{f} \\ 0 & \text{otherwise} \end{cases}$$

Note $\hat{x}$ is **feasible** according to this rounding scheme. We prove this by proving the solution according to $\hat{x}$ is a set cover, i.e. we claim for all $e \in E$, $e \in S_v$ for some v. By contradiction assume exists $e \in E$ such that $e \notin S_v$ for all v (i.e. $x_v^* < {}^1/_f$), therefore

$$\sum_{v:e\in S_v} x_v^* < \sum_{v:e\in S_v} \frac{1}{f} = \frac{f_v}{f} \leq 1$$

also note x_v^* feasible, i.e. $\sum_{v:e\in S_v} x_v^* \geq 1$, contradiction.

8. **f-approximation algorithm** Now we prove deterministic rounding above yields a f-approximation algorithm. Since $\hat{x}_v$ feasible, then we have lower bound

$$opt = Z_{IP}^* \leq \sum_j w_j \hat{x}_j$$

This *lower bound always hold for any integer LP programming that uses rounding.* Note for any $\hat{x}_v$, we have $fx_v^* \geq \hat{x}_v$ since $fx_v^* \geq 0 = \hat{x}_v$ for $x_v^* < {}^1/_f$ and $fx_v^* \geq 1 = \hat{x}_v$ for $x_v^* \geq {}^1/_f$, therefore

$$\sum_j w_j \hat{x}_j \leq \sum_j w_j (f x_j^*) = f \sum_j w_j x_j^* = f Z_{LP}^* \leq f Z_{IP}^* = f\,opt$$

therefore, $opt \leq \sum_j w_j \hat{x}_j \leq f\,opt$

9. **Dual of Relaxed LP**

$$\begin{array}{llcll}
\min & \sum_{j=1}^{n} w_j x_j & & \max & \sum_e y_e \\
s.t. & & & s.t. & \\
\forall e \in E & \sum_{j:e\in S_j} x_j \geq 1 & \xrightarrow{\text{dual of relaxed LP}} & \forall v \in V & \sum_{e:e\in S_v} y_e \leq w_u \\
\forall v \in V & x_v \geq 0 & & \forall e \in E & y_e \geq 0
\end{array}$$

By weak duality, any feasible dual solution y follows $\sum_e y_e \leq Z_{LP}^*$, therefore

$$\sum_e y_e = Z_{DLP} \leq Z_{PLP}^* \leq Z_{IP}^* = opt$$

10. **Rounding a dual solution** Let y^* be optimal solution to dual LP, and we include subset S_v such that the corresponding dual constriant is 'tight', i.e.

$$\hat{x}_v = \begin{cases} 1 & \sum_{e:e\in S_v} y_e = w_v \\ 0 & \text{otherwise} \end{cases}$$

Note we can prove $\hat{x}_v$ is feasible, i.e. collections of S_v for which $\hat{x}_v = 1$ is a set cover. proof here . General idea is that assume e not covered, then imply for all $v \in V$,

$$\sum_{e:e\in S_v} y_e < w_v$$

So we can find a smallest difference between lhs and rhs, denoted as δ, cross all dual constraints, and increment y_e by δ and obtain a solution that has better objective value than the optimal solution that we started with.

11. **f-approximation algorithm for dual rounding** Lower bound holds since $\hat{x}_v$ feasible for IP. Now we prove upper bound

$$f\,opt \geq f \sum_e y_e \geq \sum_{v\in V} \sum_{e:e\in S_v} y_e \geq \sum_{v:\hat{x}_v=1} w_v + \sum_{v:\hat{x}_v=0} 0 = \sum_v w_v \hat{x}_v$$

12. **Primal-Dual: Constructing Dual Solution** Idea is to construct a dual optimal solution by relying on complementary slackness such that we dont have to solve dual LP directly. algorithm here . General outline of primal-dual algorithm

Initialize some feasible DLP y and candidate x for PLP
while *x not feasible to PLP* **do**
  Adjust y by the slack δ, such that
    y remains feasible, dual objective increases, additional constraint become tight
  Update x according to complementary slackness condition

Idea is start with some feasible DLP variable y and use it to infer some, possibly infeasible, x to PLP

13. **Randomized Rounding** Idea is to interpret LP solution x_v^* as probability that $\hat{x}_v$ is set to 1, i.e. S_v included in the final solution with probability x_v^* for each $v \in V$ as random independent events. Let X_v be an indicator variable, $X_v = \mathbb{1}_{\hat{x}_v=1}$. Therefore $\mathsf{E}\{X_v\} = x_v^*$. Therefore we can determine expected value of the solution

$$\mathsf{E}\left\{\sum_j w_j X_j\right\} = \sum_j w_j \mathsf{P}(X_j = 1) = \sum_j w_j x_j^* = Z_{LP}^* \leq opt$$

which is a good approximation, but not every element e is covered by this procedure, the probability of a single edge e not covered is given by

$$\mathsf{P}(e \text{ not covered }) = \prod_{v:e\in S_v} (1 - x_v^*) \leq \prod_{v:e\in S_v} e^{-x_v^*} = e^{-\sum_{v:e\in S_v} x_v^*} \leq e^{-1}$$

where last inequality given by LP constraint. We want to devise a polynomial-time algorithm whose chance of failure is at most inverse of a polynomial m^{-c}, then in this case we can say the algorithm *works* with *high probability*. The **revised** algorithm works by flipping a coin that comes heads up with probability x_v^* and we flip the $c \ln m$ times and decide if we include S_v in the solution or not. Let

$$X_v = \begin{cases} 1 & \text{at least 1 head in } c\ln m \text{ coin flips with } \mathsf{P}\left(head\right) = x_v^* \\ 0 & \text{otherwise} \end{cases}$$

Note

$$\mathsf{P}\left(X_v = 0\right) = (1 - x_v^*)^{c \ln m} \qquad \mathsf{P}\left(X_v = 1\right) = 1 - (1 - x_v^*)^{c \ln m}$$

We can derive a bound on $\mathsf{P}\left(X_v = 1\right)$ by deriving its derivative $\mathsf{P}\left(X_v = 1\right)' = (c \ln m)(1 - x_v^*)^{c \ln m - 1} \leq c \ln m$ and observe that $\mathsf{P}\left(X_v = 1\right) \leq (c \ln m) x_v^*$. Now we derive probability of outputting a feasible set cover

$$\mathsf{P}\left(\text{any } e \text{ not covered}\right) = \prod_{v: e \in S_v} (1 - x_v^*)^{c \ln m} \leq \prod_{v: e \in S_v} e^{-x_v^*(c \ln m)} = e^{-(c \ln m) \sum_{v: e \in S_v} x_v^*} \leq \frac{1}{m^c}$$

Let F be event where solution is a feasible set cover, then

$$\mathsf{P}\left(\overline{F}\right) = \mathsf{P}\left(\text{exists } e \text{ uncovered}\right) \overset{unionbound}{\leq} \sum_e \mathsf{P}\left(e \text{ not covered}\right) \leq \frac{1}{m^{c-1}} \qquad \mathsf{P}\left(F\right) \geq 1 - \frac{1}{m^{c-1}}$$

We can now compute expected objective of the integer program

$$\mathsf{E}\left\{\sum_v w_v X_v\right\} = \sum_v w_v \mathsf{P}\left(X_v = 1\right) \leq \sum_v w_v (c \ln m) x_v^* = (c \ln m) Z_{LP}^* \leq (c \ln m)\, opt$$

therefore the algorithm is $O(\ln m)$-approximation algorithm

Section 1 chapter 5 : Random sampling and randomized rounding of LP

1. **MAX SAT** n boolean variables $x_1, \cdots, x_n$ and m clauses $C_1, \cdots, C_m$, each consists of a disjunction $\vee$ or some number of literals (variables and their negations) and is of length l_j, and nonnegative weight w_j for each clause C_j. Objective is to find an assignment of true/false to x_i that maximizes the weight of *satisfied* clauses. A clause is satified if the clause evaluates to true.

2. **Randomized algorithm for MAX SAT** Setting each x_i to true independently with probability $^1/_2$ gives $^1/_2$-approximation algorithm for MAX SAT problem. Let $X_j = \mathbb{1}_{C_j = 1}$, then

$$\mathsf{E}\left\{X_j\right\} = 1 \cdot \mathsf{P}\left(C_j = 1\right) = 1 - \mathsf{P}\left(C_j = 0\right) = 1 - \left(\frac{1}{2}\right)^{l_j} \geq \frac{1}{2}$$

last inequality is a loose bound by the fact that $l_j \geq 1$. Therefore

$$\mathsf{E}\left\{\sum_j w_j X_j\right\} \geq \frac{1}{2}\sum_j w_j \geq \frac{1}{2}\,opt$$

where last inequality follows from the fact that the total weight is an easy upper bound on the optimal value. In general, if $l_j \geq k$ for each clause C_j, then the algorithm becomes a $(1 - (1/2)^k)$-approximation algorithm

3. **MAX CUT** Given undirected $G = (V, E)$, $w_{ij} \geq 0$ for each $(i, j) \in E$. Objective is to partition vertex into U and $W = V \setminus U$, to maximize weight of edges whose two endpoints in different parts, i.e. edges that is *in the cut*. In case $w_{ij} = 1$ we have unweighted MAX CUT problem.

4. **Randomized algorithm for MAX CUT** If we place each $v \in V$ into U independently with probability $1/2$, the we have a $1/2$-approximation algorithm for the max cut problem. Let $X_{ij} = \mathbb{1}_{(i\in U \wedge j\in W)\vee(i\in W\wedge j\in U)}$, i.e. indicator specifing if an edge is in the cut. Note $\mathsf{E}\{X_e\} = 1/2$, then expected objective is

$$\mathsf{E}\left\{\sum_e w_e X_e\right\} = \frac{1}{2}\sum_e w_e \geq \frac{1}{2}\,opt$$

where last inequality given by the fact that optimal value bounded above by sum of weights of all edges.

5. **Derandomization** Idea is to convert a randomized algorithm to obtain a deterministic algorithm whose solution value is as good as the expected value of the randomized algorithm.

6. **Derandomization for MAX SAT** Let W be total weight of clauses for a particular assignment. Set $x_1, \cdots$ sequentially. Given we have already set $x_1, \cdots, x_i$ to $b_1, \cdots, b_i$, we next set x_{i+1} according by following

$$x_{i+1} = \begin{cases} 1 & \mathsf{E}\{W|x_1 \leftarrow b_1, \cdots, x_i \leftarrow b_i, x_{i+1} \leftarrow true\}\,\mathsf{P}(x_{i+1} \leftarrow true) \\ & \qquad > \mathsf{E}\{W|x_1 \leftarrow b_1, \cdots, x_i \leftarrow b_i, x_{i+1} \leftarrow false\}\,\mathsf{P}(x_{i+1} \leftarrow false) \\ 0 & \text{otherwise} \end{cases}$$

in other words, we set x_{i+1} that will maximize the expected value of the resulting solution. Setting it this way ensures that

$$\mathsf{E}\{W|x_1 \leftarrow b_1. \cdots, x_i \leftarrow b_i, x_{i+1} \leftarrow b_{i+1}\} \geq \mathsf{E}\{W|x_1 \leftarrow b_1. \cdots, x_i \leftarrow b_i\}$$

which is derived by expanding $\mathsf{E}\{W|x_1 \leftarrow b_1, \cdots, x_i \leftarrow b_i\}$ by laws of conditional expectation. Now by induction, implies when algorithm terminates, we have

$$\mathsf{E}\{W|x_1 \leftarrow b_1. \cdots, x_n \leftarrow b_n\} \geq \mathsf{E}\{W\} \geq \frac{1}{2}\,opt$$

therefore a $1/2$-approximation algorithm. Expectation with conditional expectation is readily computable

$$\mathsf{E}\{W|x_1 \leftarrow b_1, \cdots, x_i \leftarrow b_i\} = \sum_j w_j \mathsf{P}(C_j = 1|x_1 \leftarrow b_1, \cdots, x_i \leftarrow\leftarrow b_i)$$

$P(C_j = 1)$ is 1 if setting of $x_1, \cdots, x_i$ already satisfies the clause, and is $1 - (1/2)^k$ otherwise, where k is the number of unset literals in the clause

Section 1 chapter 7 : The primal-dual method

1. **Set Cover Problem** The algorithm

   $y \leftarrow 0$
   $I \leftarrow \emptyset$
   while $\exists e_i \notin \cup_{j \in I} S_j$ **do**
     Increase dual y_i until there is some l such that $\sum_{j:e_j \in S_l} y_j = w_l$
       $y_i \leftarrow y_i + \epsilon$ where $\epsilon = \min\limits_{j:e_i \in S_j} (w_j - \sum_{k:e_k \in S_j} y_k)$
     $I \leftarrow I \cup \{l\}$
   return I

   We want to prove the algorithm is f-approximation algorithm where $f = \max\limits_i |\{j : e_i \in S_j\}|$, i.e. max number of times an edge appears in different subsets. In any primal-dual algorithm we have

   $$f \cdot \sum_{i=1}^n y_i \le f \cdot Z^*_{LP} \le f \cdot opt$$

   Note for any $j \in I$, the dual constraint is satisfied, i.e. $w_j = \sum_{i:e_i \in S_j} y_i$, therefore

   $$\sum_{j \in I} w_j = \sum_{j \in I} \sum_{i:e_i \in S_j} y_i = \sum_{i=1}^n y_i \cdot |\{j \in I : e_i \in S_j\}| \le f \sum_{i=1}^n y_i \le f \cdot opt$$

2. **Standard primal-dual analysis**:

   (a) Maintain a feasible dual
   (b) Increase dual variables until a dual constraint becomes tight. This indicates we need to add to our primal solution.
   (c) When analyze cost of primal solution, each object in the solution was given by a tight dual inequality ($A^T y = c$) and so we can rewrite cost of primal solution $c^T x$ in terms of dual variables $y^T Ax$.
   (d) Then we compare this cost with dual objective function and show that the primal cost is within a certain factor of the dual objective.

In summary, the standard primal-dual algorithm constructs a primal integer solution and a solution to the dual of the linear programming relaxation.

3. **Feedback Vertex Set Problem** Given undirected graph $G = (V, E)$, and vertex weights $w_i \geq 0$ for all $i \in V$. Goal is to choose min-cost subset of vertices $S \subseteq V$ such that every cycle C in the graph contains some vertex of S, i.e. S *hits* every cycle of the graph. Alternatively, we want to find minimum-cost subset S such that removing S leaves G acyclic, i.e. the induced graph $G[V \setminus S]$ is acyclic.

4. **Induced Graph** $G[V \setminus S]$ is an induced graph on $V \setminus S$ with edges from G that have both endpoints in $V \setminus S$

5. Integer program and the dual of relaxed linear program

$$
\begin{array}{rll}
\min & \sum_{v\in V} w_v x_v \\
s.t. & \\
\forall C \in \mathcal{C} & \sum_{v\in C} x_v \geq 1 \\
\forall v \in V & x_v \in \{0,1\}
\end{array}
\qquad
\begin{array}{rll}
\max & \sum_{C: C\in\mathcal{C}} y_C \\
s.t. & \\
\forall v \in V & \sum_{C\in\mathcal{C}: v\in C} y_C \leq w_v \\
\forall C \in \mathcal{C} & y_C \geq 0
\end{array}
$$

the primal-dual algorithm given by

$y \leftarrow 0$
$S \leftarrow \emptyset$
while *exists cycle C in G* **do**
  Find cycle C with at most $2\lceil \log_2 n \rceil$ vertices of degree 3 or more with bfs
  Increase y_C until there is some $l \in C$ such that $\sum_{C'\in\mathcal{C}: l \in C'} y_{C'} = w_l$
    $y_C \leftarrow y_C + \epsilon$ where $\epsilon = \min_{i\in C}(w_i - \sum_{C': i\in C'} y_{C'})$
  $S \leftarrow S \cup \{l\}$
  Remove l from G
  Repeated remove vertices of degree one from G
return S

Idea is since we can remove at most n vertices, we only need to maintain $|S| \leq n$ as primal solution, versus the potentially exponential number of dual variables. We now analyze the algorithm. Let S be final set of vertices chosen, know for all $v \in S$, $w_v = \sum_{C: v\in C} y_C$. So the cost of IP solution is given by

$$\sum_{v\in S} w_V = \sum_{v\in S} \sum_{C: v\in C} y_C = \sum_{C\in\mathcal{C}} |S \cap C| y_C$$

where $|S \cap C|$ is simply the number of vertices of the solution S in the cycle C. Idea is we want to bound $|S \cap C| \leq \alpha$, but for arbitrary cycle, $|S \cap C|$ can be quite large. Idea is we want to pick smaller cycles with some bound such that $|S \cap C| \leq |C| \leq \alpha$

6. **Picking the Right Cycle**

   (a) For any path P consisting over vertices of degree two in G. The algorithm choose at most one vertex from P, i.e. $|S \cap P| \leq 1$ for the final solution S given by the algorithm

   (b) In any graph G with no vertices of degree one, there is a cycle with at most $2\lceil \log_2 n \rceil$ vertices of degree three or more, and it can be found in linear time.

   Proof. There is a cycle since acyclic graph with n vertices has at most $n-1$ edges. But $2m = \sum_u deg(u) \geq 2n$, so must exists a cycle. Convert G to H where we treat every path of vertices of degree two as a single edge joining vertices of degree 3 or more. If we run bfs on H, there is at least twice the number of vertices in successive level, so the number of levels is bounded $l \leq \lceil \log_2 n \rceil$. We can find a path of vertices of degree two from a node on the current level to a previously visited node; This forms a cycle with at most $2\lceil \log_2 n \rceil$ vertices of degree two or more. Therefore we have found a cycle with at most $2\lceil \log_2 n \rceil$ vertices of degree three or more in G, and the algorithm terminates in $O(m+n)$. □

   Iea is can find a cycle C that minimies number of vertices having degree 3 or higher, i.e. a cycle C with $2\lceil \log_2 n \rceil$ vertices of degree 3 or higher. In the worst case, vertices of degree 3 or more alternates with paths of vertices of degree two, so $|S \cap C| \leq 4\lceil \log_2 n \rceil$

7. $(4\lceil \log_2 n \rceil)$**-approximation algorithm** Idea is $y_C > 0$ only if C contains at most $2\lceil \log_2 n \rceil$ vertices of degree three or more. Claim if C has at most $2\lceil \log_2 n \rceil$ vertices of degree three or more in C can contain at most $4\lceil \log_2 n \rceil$ vertices of S overall

   (a) possibly degree 3 vertex is in S

   (b) at most one of vertices in the path joining adjacent vertices of degree 3 or more is in S

   Therefore whenever $y_C = 0$, we have $|S \cap C| \leq 4\lceil \log_2 n \rceil$, so

$$\sum_{v \in S} w_v = \sum_{c \in \mathcal{C}} |S \cap C| y_C \leq (4\lceil \log_2 n \rceil) \sum_{C \in \mathcal{C}} y_C \leq (4\lceil \log_2 n \rceil) \cdot opt$$

   Important observation is that to get good performance guarantee, one must carefully pick the dual variable to increase, i.e. pick a dual variable that is small or minimial in some sense