Lecturer: Anupam Gupta Scribe: Hang Liao and Zhen Zhou

1 Approximation Algorithms: Overview

In this lecture, we'll go through approximation algorithm for two NP-hard problems: Set Cover and Bin-packing.

Before going into the algorithm, let's make a few definitions and notations to help clarify the terminology. Denote $c(A(I))$ as the general cost of that algorithm A gives for instance I, and denote $c(OPT(I))$ as the optimal cost for instance I. In the context of this lecture, we are only looking at minimization problems. Therefore, we can define the approximation ratio for minimization problems as:

$$r = \max_I \frac{c(ALG(I))}{c(OPT(I))}$$

For the rest of the lecture notes, we'll use informal notations like

$$ALG = r \cdot OPT$$

to denote that the algorithm ALG has an approximation ratio of r. We'll also say that ALG is an r approximation algorithm.

Approximation algorithm has been a long standing topic in theoretical computer science, it is usually used to get solutions that are relatively good compared to optimized solution. As an overview, we can classify the approximation problems by hardness: (For the following, assume OPT is the best algorithm for every instance)

- Problems in P: Fully Poly-Time Approximation Scheme (FPTAS): For problems in this category, $\forall \epsilon > 0$, there exists an approximation algorithm $\mathcal{A}$, with approximation ratio $r = 1 + \epsilon$, that runs in time poly$(|I|, \frac{1}{\epsilon})$.

  (Note: If an algorithm $\mathcal{A}$ has an approximation ratio $r = 1 + \epsilon$, then for all instances I, $c(\mathcal{A}(I)) \leq (1 + \epsilon)c(OPT(I))$.)

- Poly-Time Approximation Scheme (PTAS): This category includes problems such as the Traveling Salesman Problem in Euclidean space. For any problem in this category, there exists an approximation algorithm $\mathcal{A}$, with approximation ratio $r = 1 + \epsilon$, that runs in time $O(n^{f(\epsilon)})$ for some function $f(\epsilon)$.

- Constant Factor Approximation: Example: Traveling Salesman Problem (metric TSP). In 1976 and 1978, Christofides [Chr76] and Serdyukov [Ser78] discovered the same 1.5 approximation algorithm separately for metric TSP. Sebő and Vygen discovered a 1.4 approximation algorithm in the context of unweighted graphs in 2012 [SV12].

  Meanwhile, it has been shown that metric TSP can't be approximated with a ratio better than $\frac{123}{122}$ under the assumption of P $\neq$ NP (by Karpinski, Lampis, Schmied) [KLS15].

- Log Factor Approximation: Example: Set Cover, as we'll see later.

- Non-approximable: Example: Independent Set. In fact, any algorithm with an approximation ratio $r \leq n^{1-\epsilon}$ for some constant $\epsilon > 0$ will lead to the conclusion P = NP. The best approximation algorithm for Independent Set so far has an approximation ratio of $r = \frac{n}{\log^3 n}$

However, there are some problems that don't fall into any of these categories, such as the problem of Asymmetric K Center, there exists a $\log^* n$ approximation algorithm.

2 Surrogate and LP

Since it is usually hard to find the optimal solution, how do we argue that some algorithm A has approximation ratio C? An important idea in proving approximation ratio involves the use of a surrogate.

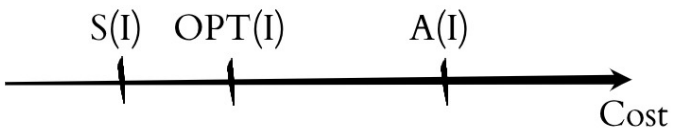

Figure 23.1: The cost diagram on instance I (cost increase from left to right)

Given an approximation algorithm $\mathcal{A}$ and an instance I, we want to calculate the approximation ratio of $\mathcal{A}$. We can do so by finding a surrogate S. The typical trick of proving approximation ratio is to first show $S(I) \leq OPT(I)$, then show that $\mathcal{A}(I) \leq \alpha S(I)$, and therefore $\mathcal{A}(I) \leq \alpha OPT(I)$. If we can find a surrogate S that satisfies this property for all instances I, and prove the relation $\mathcal{A}(I) \leq \alpha S(I)$ for all instances I, then we are able to prove $\mathcal{A}$ has an approximation ratio of at most α.

A common question is "how to construct the surrogate". As we'll see, the surrogate is usually a relaxed LP.

3 Set Cover

The Set Cover problem can be formulated as following: Given a universe U of n elements, and $\mathscr{S}$, a family of m subsets of U, such that $U = \bigcup_{X \in \mathscr{S}} X$. We want to find a subset $P' \subseteq P$, such that $U = \bigcup_{X \in P'} X$ while minimizing $|P'|$.

Set Cover is NP-complete. In a more generalized version of Set Cover, we can assign cost c_X to each set $X \in \mathscr{S}$, then we want to minimize $c(P') = \sum_{X \in P'} c_X$. In the generalized setting, the problem is still NP-hard. But for now, we'll only look at the case where we want to minimize $|P'|$, which is equivalent to setting the cost of each set X to 1.

There are a few approximation algorithms of Set Cover. The greedy algorithm we show below is an $O(\log n)$ approximation algorithm due to Chvátal[Chv79], Johnson[Joh74], Lovász[Lov75], Stein [Ste74] from 1974 to 1979. Furthermore, in 1998, Feige showed that if there exists an $(1-\epsilon)\ln n$ approximation algorithm for any constant $\epsilon > 0$, then P = NP[Fei98]. Dinur and Steurer have also shown results in NP-hardness of approximation of Set Cover.[DS14]

We'll be showing two algorithms, the first algorithm is a simple greedy algorithm as follows:

3.1 Greedy Algorithm for Approximating Set Cover

- Greedy Algorithm (Set Cover): Pick the set S that covers the most uncovered elements. Repeat until we've covered all elements of U.

Theorem 23.1. *The greedy algorithm is a* $\ln n + 1$ *approximation algorithm.*

Proof. Suppose OPT picks k sets from $\mathscr{S}(k \geq 1)$.

First, let's try to convince ourselves that the greedy algorithm can't achieve a better ratio than $\log n$. An intuitive example is when the universe is a set of 18 dots, and we have following sets of points $P_1, P_2, S_1, S_2, S_3, S_4$.

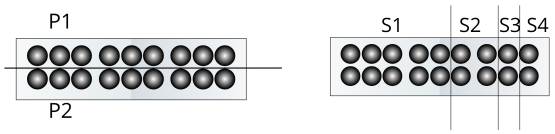

Figure 23.2

Notice that by our greedy algorithm, we'll pick S_1 first, then possibly we'll pick S_2, S_3, S_4(suppose we make unfortunate choices when there is a tie). Then we picked 4 sets for greedy algorithm. However, notice that the optimal algorithm will pick P_1, P_2 so it covers all 18 dots with just 2 sets. In general, if we have n dots in this problem, our greedy algorithm will pick $O(\log n)$ sets, where the optimal algorithm will just pick P_1 and P_2, a total of 2 sets. This shows that our approximation ratio is at least $O(\log n)$.

Then, we want to show that the greedy algorithm is indeed an $\ln n + 1$ approximation algorithm. Let $Uncov(i)$ denote the number of points remaining at time i. Naturally, we have $Uncov(i) = n$. At any time T, the number of remaining elements is $Uncov(T)$, notice that the remaining elements can be covered with at most k sets, since the universe can be covered with k sets.

Therefore, there must exist some set S not chosen that covers at least $\frac{1}{k}$ fraction of the remaining elements. Since greedy algorithm picks a set no worse than S, it follows directly that $Uncov(T+1) \leq (1 - \frac{1}{k}) \cdot Uncov(T)$. Applying this relation recursively to get:

$$\begin{aligned} Uncov(T) &\leq (1 - \frac{1}{k})^T n_0 \\ &= (1 - \frac{1}{k})^T n \quad (\text{since } n_0 = n) \\ &\leq n e^{\frac{-T}{k}} \end{aligned}$$

Notice that if $T \geq k \ln n + 1$, then $Uncov(T) \leq e^{\frac{-T}{k}} n \leq \frac{1}{e} < 1$, which means that we've covered all the elements. Therefore, it shows that the greedy algorithm picks at most $k \ln n + 1$ sets. Since $k \geq 1$, we know that it is also true that greedy algorithm picks at most $k(\ln n + 1)$ sets. This shows that greedy algorithm is an $\ln n + 1$ approximation. □

3.2 LP Algorithm for Approximating Set Cover

The second algorithm uses LP and surrugate LP:

- LP Algorithm (Set Cover): Let's have m unknowns, each unknown x_S for a set $S \in \mathscr{S}$. Construct the following LP (surrogate LP):

$$\begin{aligned} &\min \sum_{S \in \mathscr{S}} x_S \\ &s.t. \sum_{S:e \in S} x_S \geq 1, \forall e \in U \\ &x_S \geq 0, \forall S \in \mathscr{S} \end{aligned}$$

  Solve this LP optimally to get a solution x^*, then run the following two phases:

  Phase 1: Repeat the following for $t = \ln n$ times: For all set S, pick S with probability x_S^*.

  Phase 2: If there exist any covered element e, pick any set that covers it.

Proof. First consider the simplified version of LP(in fact, ILP):

$$\begin{aligned} &\min \sum_{S \in \mathscr{S}} x_S \\ &s.t. \sum_{S:e \in S} x_S \geq 1, \forall e \in U \\ &x_S \in \{0,1\}, \forall S \in \mathscr{S} \end{aligned}$$

Notice that the solution to this LP (ILP) is exactly OPT. Since the value in the original LP can be fractional, solving the original LP (surrogate) will give us a target function value no more than ILP, and therefore no more than OPT.

First consider the expected number of elements not covered in Phase 1: Specifically, fix an element e. The probability of e not being covered in Phase 1 is as follows:

$$\begin{aligned} Pr[\text{e not covered}] &= (\Pi_{S:e \in S}(1 - x_s^*))^t \\ &\leq (e^{\sum_{S:e \in S} x_S})^t \\ &\leq (e^{-1}) \\ &= e^{-t} \\ &= \frac{1}{n} \end{aligned}$$

Therefore, by linearity of expectation, the expected number of uncovered elements in Phase 2 should be 1. So in expectation we'll pick 1 set in Phase 2. Then consider total number of sets picked: (Let LP_{val} denote the optimized target function value in LP.)

$$\begin{aligned} E[\text{number of sets picked}] &\leq (\sum_S x_S^*) \ln n + n \cdot \frac{1}{n} \\ &= LP_{val} \cdot (\ln n) + 1 \\ &\leq (\ln n + 1) LP_{val} \\ &\leq (\ln n + 1) OPT \end{aligned}$$

Therefore, in expectation, the LP algorithm is a $\ln n + 1$ approximation. We can get rid of this expectation by derandomizing the last two phases. □

4 Bin Packing

Bin Packing is another classic NP-hard problem. Suppose we have n items, each item i with size $s_i \in [0,1]$. We want to find the minimum number of bins, each with capacity 1, that can pack all n items. Formally, given $s_1, s_2, ..., s_n \in [0,1]$, we want to find the partition of $\{1,2,...,n\} = S_1 \cup S_2 \cup ... \cup S_k$ such that $\forall j \in [k], \sum_{i \in S_j} s_i \leq 1$ with k minimized.

We will provide two algorithms. The first algorithm First Fit uses at most $2 \cdot OPT + 1$ bins and the second algorithm (due to Fernandez de la Vega and Lueker [dlVL81]) uses at most $(1+\epsilon) \cdot OPT + O(\frac{1}{\epsilon^2})$ bins. A recent result by Rebecca Hoberg and Thomas Rothvoss [HR17] gives a solution using at most $OPT + O(\log OPT)$ bins.

4.1 Greedy: First Fit

A natural idea is to assign each item starting from the first bin until it fits.

Theorem 23.2. *Let ALG_{FF} denote the number of bins used by First Fit. Then*

$$ALG_{FF} \leq 2 \cdot OPT + 1$$

Proof. We call a bin "half-full" if at least half of the bin is filled. Note with First Fit algorithm all but one bin must be half-full. This follows by induction on item index. The base case is true as only one bin is used when only item 1 is packed. For the inductive step, consider the last item n we put into the bins.

- If $s_n \geq \frac{1}{2}$, we are done as no matter which bin item n is packed into, there can't be more bins that are not half-full.
- If $s_n < \frac{1}{2}$ and originally there was one bin not half-full, item n will be packed into the bin by First Fit principle.
- If $s_n < \frac{1}{2}$ and originally all bins were half-full, the worst case after item n arrives is one bin not half-full.

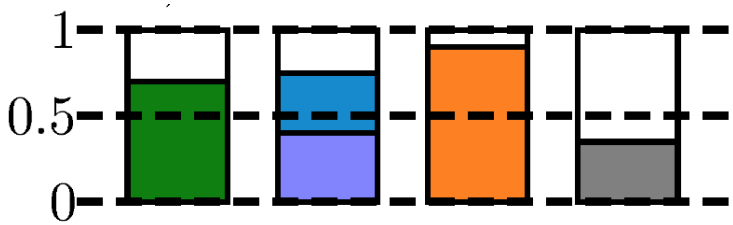

Figure 23.3: at most one bin is not half-full

Therefore at least $ALG_{FF}-1$ bins are half-full. We have $\frac{1}{2}(ALG_{FF}-1) \leq \sum_{i=1}^{n} s_i$. Notice that even if we pack our items optimally (imagine OPT can pack items like they are sand!), $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$.

Combine the inequalities we have $\frac{1}{2}(ALG_{FF} - 1) \leq OPT \implies ALG_{FF} \leq 2 \cdot OPT + 1$. □

4.2 Partition is NP-hard

The Partition problem is NP-hard. Given n positive integers $s_1, s_2, ..., s_n$ and an integer k such that $\sum_{i=1}^{n} = 2K$. Partition is the task of finding a partition of $\{1,2,...,n\}$ into two disjoint sets A, B such that $\sum_{i \in A} s_i = \sum_{j \in B} s_j = K$.

4.3 Bin Packing and Partition

Claim 23.3. *An α-approximation algorithm for Bin Packing with $\alpha < \frac{3}{2}$ solves Partition.*

Proof. Given an instance of Partition. Normalize every integer by K. Now the original Partition problem is reduced to a Bin Packing problem. "There exists a partition" if and only if "there exists a 2 Bin Packing". By the α-approximation algorithm, we can find the 2 Bin Packing in poly-time as the number of bins returned by the algorithm is always integer and $2 = \lfloor 2\alpha \rfloor \leq 2\alpha < 3$.

So the α-approximation algorithm is in fact 1-approximation algorithm for Partition. □

4.4 Algorithm by Fernandez de la Vega & Lueker

Lemma 23.4. *(Linear Grouping) Given an instance of Bin Packing $I = (s_1, s_2, ..., s_n)$ and $D \in \mathbb{N}$, there exists $I' = (s_1', s_2', ..., s_n')$ with $s_i' > s_i$ and at most D different item sizes that satisfies $OPT(I') \leq OPT(I) + \frac{n}{D}$.*

Proof. The instance I' is constructed as follows:

- Sort s_i to get $s_1 \leq s_2 \leq ... \leq s_n$.
- Partition items into D consecutive groups of $\lceil \frac{n}{D} \rceil$ items. The last group is allowed to have less items.
- Let s_i' be the size of the largest element in i's group.

Suppose OPT assigns s_i to some bin b. Then we assign ${s'}_{i-\lceil \frac{n}{D} \rceil}$ to bin b. By construction, ${s'}_{i-\lceil \frac{n}{D} \rceil} \leq s_i$. So items can fit into the bins. The strategy takes care of all the items except items in last group. We assign each item in the last group into a new bin. The assignment costs at most $\lfloor \frac{n}{D} \rfloor$ bins as the last group is allowed to have less items.

With this particular algorithm on instance I', the total number of bins used is at most $OPT(I) + \lfloor \frac{n}{D} \rfloor$. Therefore $OPT(I') \leq OPT(I) + \frac{n}{D}$. □

Theorem 23.5. *The algorithm by Fernandez de la Vega & Lueker costs at most $(1+\epsilon) \cdot OPT + O(\frac{1}{\epsilon^2})$ bins.*

Proof. We make two assumptions that are going to be removed later:

1. Every item has size at least ϵ.
2. The number of different item sizes is at most D.

An idea by Gilmore and Gomory [GG61] is to consider the patterns of item sizes in each bin. With the two assumptions, there can be at most $\frac{1}{\epsilon}$ items in each bin and each item has D possible sizes. The number of distinct pattern is at most $D^{\frac{1}{\epsilon}}$. ($(D+1)^{\frac{1}{\epsilon}}$?)

Define x_p to be the number of copies with pattern p. Let p_s be the number of size s items in pattern p, and $|s|$ be the number of size s items. Note p_s and $|s|$ are constants. We can write an integer LP to capture the Bin Packing problem as follows:

$$\min \sum_p x_p,$$
$$s.t.\ \forall s, \sum_p x_p p_s \geq |s|,$$
$$x_p \in \mathbb{N}$$

A natural relaxation of the above LP (the surrogate):

$$\min \sum_p x_p,$$
$$s.t.\ \forall s, \sum_p x_p p_s \geq |s|,$$
$$x_p \geq 0$$

The LP had at most D non-zero variables. Observe that the LP had a large number of variables (to be precise, $D^{\frac{1}{\epsilon}}$; lets call this number N) each corresponding to some pattern, but only D non-trivial constraints, and the rest N were all non-negativity constraints. Consider a basic solution. By definition, there are N linearly independent tight constraints at this point. Only D of these can be non-trivial. So the other $N - D$ tight constraints must each say $x_i = 0$ for some i. This means $N - D$ variables are zero. And at most D variables are non-zero, as claimed.

Rounding the variables up gives us an integer solution with at most $LP + D \leq OPT + D$ bins.

Remove assumption 2:

Define VOL to be the sum of item sizes. With assumption 1 we have

$$\epsilon n \leq VOL \leq OPT \tag{23.1}$$

Apply Lemma 4.4 with $D = \frac{1}{\epsilon^2}$. $\frac{n}{D} = \epsilon^2 n \leq \epsilon \cdot OPT$. This implies the second assumption can create additional cost at most $\epsilon \cdot OPT$ bins.

Remove assumption 1: Use First Fit to pack the small items. Two cases:

1. No new bin is needed. Great!

2. New bins are opened. Consider the last bin opened. The bin is opened because all the previous bins are at least $(1 - \epsilon)$ full, otherwise by First Fit we won't open this bin. So the number of bins opened is at most $\frac{1}{1-\epsilon} \cdot OPT + 1 \leq (1 + 2\epsilon) \cdot OPT + 1$ given $\epsilon \in (0, 1)$.

The linear program gives us a solution with $OPT + D$ bins. To remove the two assumptions we need $\epsilon \cdot OPT + 2\epsilon \cdot OPT + 1$ bins. The total number of bins with the algorithm is $(1+3\epsilon) \cdot OPT + \frac{1}{\epsilon^2} + 1 = (1 + \epsilon') \cdot OPT + O(\frac{1}{\epsilon'^2})$. □

Acknowledgments

These lecture notes were scribed by Hang Liao and Zhen Zhou. Figure 23.3 is from course slides by Thomas Rothvoss.

References

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976. 1

[Chv79] Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979. 3

[dlVL81] Fernandez de la Vega and Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981. 4

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014. 3

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. 3

[GG61] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9(6):849–859, 1961. 4.4

[HR17] Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2616–2625, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. 4

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974. 3

[KLS15] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015. 1

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975. 3

[Ser78] A Serdyukov. On some extremal walks in graphs. *Upravlyaemye systemy*, 17:76–79, 1978. 1

[Ste74] S. K. Stein. Two combinatorial covering theorems. *J. Comb. Theory, Ser. A*, 16(3):391–397, 1974. 3

[SV12] András Sebö and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for graphic tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *CoRR*, abs/1201.1870, 2012. 1