Hint of Assignment 8: Approximation Algorithms

Chao Wang (janiccordan@gmail.com)
Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

1. Bin Packing

Given a bin S of size V and a list of n items with sizes $a_1, \ldots, a_n$ to pack, find an integer number of bins B and a B-partition $S_1 \cup \ldots \cup S_B$ of set $\{1, \ldots, n\}$ such that

$$\sum_{i \in S_k} a_i \leqslant V$$

for all $k = 1, \ldots, B$. A solution is optimal if it has minimal B. Give a 2-approximation algorithm for this problem. And prove the following claim: For any $\epsilon > 0$, there is no approximation algorithm having a guarantee of $3/2 - \epsilon$ for the bin packing problem, assuming $P \neq NP$.

Hint:
Consider the algorithm called First-Fit. This algorithm considers items in an arbitrary order. In the ith step, it has a list of partially packed bins, say $B_1, \ldots, B_k$. It attempts to put the next item, a_i, in one of these bins, in this order. If a_i does not fit into any of these bins, it opens a new bin B_{k+1}, and puts a_i in it. If the algorithm uses m bins, then at least $m-1$ bins are more than half full, otherwise at least 2 bins are no more than half full, implies these 2 bins can merged into 1 bin. Therefore,

$$\sum_{i=1}^{n} > \frac{m-1}{2}$$

Since the sum of the item sizes is a lower bound on OPT, $m - 1 < 2OPT$, i.e., $m \leqslant 2OPT$.
If there exists an algorithm with guarantee of $3/2 - \epsilon$, then we show how to slove the NP-hard problem of deciding if there is a way to partition n nonnegative numbers $a_1, \ldots, a_n$ into two sets, each adding up to $\frac{1}{2}\sum_i a_i$. Clearly, the answer to this question is 'yes' iff the n items can be packed in 2 bins of size $\frac{1}{2}\sum_i a_i$. If the answer is 'yes' the $3/2 - \epsilon$ factor algorithm will have to give an optimal packing, and thereby solve the partitioning problem.

2. Vertex Cover

Consider the following algorithm for unweighted : In each connected component of the input graph execute a depth first search (DFS). Output the nodes that are not the leaves of the DFS tree. Show that the output is indeed a vertex cover, and that it approximates the minimum vertex cover within a factor of 2.

Hint:
Let T be the DFS tree, L be the set of leaves, n be the total number of vertices. To see that $V - L$ is a vertex cover, note that in the DFS tree, all edges are either forward edges which go from ancestor to descendant or back edges which go from descendants to ancestors. Thus there is no edges between leaves, and hence $V - L$ is a valid vertex cover.

To see that its within two times the optimal, note that any vertex cover of the graph must also be a vertex cover of the tree. Note, for a tree we can assume that none of the leaves are in the optimal vertex cover, it is always more benefecial to take the parent if a leaf is in the cover. For any vertex cover VC we have

$$\sum_{v \in VC} deg(v) \geqslant |E|$$

since it must cover every edge. Also we have

$$\sum_{v \in VC} deg(v) + \sum_{v \notin VC} deg(v) = 2|E|$$

which gives us

$$\sum_{v \notin VC} deg(v) \leqslant |E|$$

Since all the leaves are out of the vertex cover, and for any nonleaf the degree is at least 2, we get, $2|E| \geqslant |L| + 2(n - |L| - VC)$ which on rearranging gives us $VC \geqslant \frac{n-|L|+1}{2}$. Thus, for a tree any vertex cover must have these many vertices, which is more than half of the vertices returned by the algorithm, proving the claim of 2-approximation.

3. 3D Matching

Consider the following maximization version of the 3-Dimensional Matching Problem. Given disjoint sets X, Y, Z, and given a set $T \subseteq X \times Y \times Z$ of ordered triples, a subset $M \subseteq T$ is a 3-dimensional matching if each element of $X \cup Y \cup Z$ is contained in at most one of these triples. Assume that $|X| = |Y| = |Z|$. The Maximum 3-Dimensional Matching Problem is to find a 3-dimensional matching M of maximum size. Give a polynomial-time algorithm that finds a 3-dimensional matching of size at least 1/3 times the maximum possible size.

Hint:
A greedy iterative algorithm works. In each iteration, select an element t in T arbitrarily randomly, removed t, together with all the elements which is conflicted with t, that is the elements which x dimension is the same with t, or y dimension, or z dimension. The iteration stops when T is empty. The greedy iterative algorithm outputs all the ts during iteration. And also the size is just the number of iterations. Obviously this output is valid, since all elements which can contain conflict are removed in advance. Besides, OPT is no more than 3 times of the number or iterations, since in each iteration, OPT contains at most 3 elements. This claim can be proved easily by contradiction.

4. Hitting Set

Consider an optimization version of the Hitting Set Problem defined as follows. We are given a set $A = a_1, a_2, \ldots, a_n$ and a collection $B_1, B_2, \ldots, B_m$ which are subsets of A. Also, each element $a_i \in A$ has a weight $w_i \geqslant 0$. The problem is to find a hitting set $H \subseteq A$ such that the total weight of the elements in H, that is,

$$\sum_{a_i \in H} w_i$$

is as small as possible. Notice that H is a hitting set if $H \cap B_i$ is not empty for each i. Let $b = max_i|B_i|$ denote the maximum size of any of the sets $B_1, B_2, \ldots, B_m$. Give a polynomial-time approximation algorithm for this problem that finds a hitting set whose total weight is at most b times the minimum possible.

Hint:
When you transform the relaxed LP solution to original ILP solution, you should take $1/b$ as the rounding bound, in order to make sure all the constrains hold.

5. Programming 1: LP+Rounding

The MAX-3SAT is a problem in the computational complexity subfield of computer science. It generalizes the Boolean satisfiability problem which is a decision problem considered in complexity theory. It is defined

as: Given a 3 conjunctive normal formula ϕ (i.e. with at most 3 variables per clause), find an assignment that satisfies the largest number of clauses.

Formulate this problem under ILP and then use the LP-based rounding technique to derive an approximation algorithm for it. Achieve it in any programming language.

Hint:
Denote $x_1, \ldots, x_n$ as the variables in MAX-3SAT problem. A clause in MAX-3SAT, such as

$$x_1 \wedge x_3 \wedge \bar{x_6}$$

can be achieved by

$$x_1 + x_3 + (1 - x_6) \geqslant 1$$

in the constrains of ILP.

The binary variables in ILP is relaxed as real variables between $[0, 1]$ in LP.

6. Programming 2: PTAS

Recall the PTAS algorithm of KNAPSACK problem of Lecture 11. Achieve it in any programming language.