

1. [Next-fit strategy for BIN-PACKING] Keep on adding the items to the most recently opened bin so long as possible. When the insertion of the next item lets the capacity of the current bin exceed, close the current bin, and open a new bin.

(a) Prove that this is a 2-approximation algorithm.

(a) Let m be the number of bins used by the next-fit strategy. Suppose that a bin j is at most half full in this packing. But then, the next bin $j + 1$ must be more than half full. Otherwise, the opening of bin $j + 1$ is not justified. More importantly, the total weight of the two bins j and $j + 1$ must be larger than the bin capacity C . In fact, the sum of the weights of the items placed in bin j and the weight of the first item put in bin $j + 1$ must be larger than C . It follows that among the first $m - 1$ bins, at most $m/2$ bins can be at most half full, each followed by a bin more than half full that makes the average weight of the two consecutive bins larger than $C/2$. Therefore the total weight packed is $\sum_{i=1}^n a_i > Cm/2$. Moreover, $C \times \text{OPT} \geq \sum_{i=1}^n a_i$, that is, $m < 2 \times \text{OPT}$. Since m and OPT are integers, we have $m \leq 2 \times \text{OPT} - 1$.

(b) Prove that the approximation ratio of 2 is tight, that is, given any $\varepsilon > 0$, there exists a collection for which the approximation ratio is $> 2 - \varepsilon$.

(b) Given $\varepsilon > 0$, choose a positive integer $k > \frac{1}{\varepsilon}$. Let there be $k - 1$ items of weight k , and k items of weight 1, and $C = k$. Suppose that these items occur in the sequence $1, k, 1, k, \dots, 1, k, 1$. We have $\text{OPT} = k$ (place all the objects of weight 1 in one bin, and each item of weight k in a single bin; this strategy must be optimal since all the k bins are filled to the capacity). The next-fit algorithm uses $m = 2k - 1$ bins. The approximation ratio is therefore $2 - \frac{1}{k} > 2 - \varepsilon$.

2. [Linear BIN-PACKING] The items must be packed in the bins in the order they appear in the input. Prove that the next-fit strategy solves the linear bin-packing problem exactly.

G_{RD} — greedy solution (next-fit)

OPT — optimal solution

Change OPT to G_{RD} without
degrading the objective

OPT	s_1	s_2	s_3	s_4	$t_4 > s_4$
-----	-------	-------	-------	-------	-------------

G _{RD}	s_1	s_2	s_3	t_4
-----------------	-------	-------	-------	-------

3. Assuming that $P \neq NP$, prove that the BIN-PACKING problem cannot have a ρ -approximation algorithm for any $\rho < 3/2$.

Consider the reduction from PARTITION to BIN-PACKING in the solution of Exercise 6.22. Suppose that there exists a ρ -approximation algorithm for BIN-PACKING for $\rho < 3/2$. If two bins suffice (that is, the partitioning problem has a solution), then the algorithm will output the number of bins as $m < (3/2) \times 2 = 3$. But m is an integer, so the output will be $m = 2$. If two bins do not suffice, we have $m \geq 3$ anyway. Therefore the hypothetical algorithm for BIN-PACKING solves the partition problem in polynomial time, a contradiction to the NP-Completeness of PARTITION.

4. An algorithm is called pseudo-polynomial-time if it runs in time polynomial in the size of the input expressed in unary. An NP-Complete problem is called weakly NP-Complete if it admits a pseudo-polynomial-time algorithm. For example, we have seen that the KNAPSACK problem is weakly NP-Complete.

Prove that the SUBSET-SUM problem is weakly NP-Complete.

Let a_1, a_2, \dots, a_n be the integers and t the target sum in an instance of SSP. Let $A = \sum_{i=1}^n a_i$. The unary size of this input is $O(A + n)$ (assume that $t \leq A$). Thus, we need to have an algorithm with running time polynomial in both n and A . Here goes one.

Build an $(n + 1) \times (A + 1)$ table T such that $T(i, j)$ is the decision of SSP on a_1, a_2, \dots, a_i, j . The table is populated in the row-major order. The zeroth row is initialized as

$$T(0, j) = \begin{cases} 1 & \text{if } j = 0, \\ 0 & \text{if } j > 0. \end{cases}$$

Here, $i = 0$ means there are no input integers a_i , so the only sum achievable is 0.

Subsequently, for $i \geq 1$, we consider two cases. If $j < a_i$, then we cannot include a_i to achieve a sum of j , that is, a sum of j is achievable if and only if the first $i - 1$ integers have a subcollection of sum j . On the other hand, if $j \geq a_i$, then we have a choice of including a_i in a subcollection. If we include a_i , we check whether the remaining sum $j - a_i$ can be achieved by a subcollection of the first $i - 1$ integers. If we do not include a_i , then the sum j itself has to be achieved by a subcollection of a_1, a_2, \dots, a_{i-1} . To sum up, we have

$$T(i, j) = \begin{cases} T(i - 1, j) & \text{if } j < a_i, \\ T(i - 1, j - a_i) \text{ OR } T(i - 1, j) & \text{if } j \geq a_i. \end{cases}$$

Finally, we return $T(n, t)$ as the output.

This algorithm needs to compute $\leq (n + 1)(A + 1)$ entries in the table T with each entry requiring only $O(1)$ time (better $O(\log n + \log A)$ time, since i can be as large as n and j as large as A). It follows that the running time is polynomial in both n and A , as desired.

5. Propose an LP formulation of the BIN-PACKING problem.

Clearly, n bins suffice for the bin-packing problem. We use n Boolean variables x_j , $j = 1, 2, \dots, n$, to indicate whether the j -th bin is used. We also use n^2 variables $y_{i,j}$ to indicate whether the i -th object goes to the j -th bin. Since each object can go to exactly one bin, we have the constraints

$$\sum_{j=1}^n y_{i,j} = 1 \text{ for all } i = 1, 2, \dots, n.$$

Also, the weight capacity of each bin must be respected. This gives the constraints

$$\sum_{i=1}^n w_i y_{i,j} \leq C x_j \text{ for all } j = 1, 2, \dots, n.$$

Finally, if one wants bins with smaller numbers to fill up earlier than those with larger numbers, one may add the constraints

$$x_1 \geq x_2 \geq x_3 \geq \dots \geq x_n.$$

The objective is to minimize $\sum_{j=1}^n x_j$.

6. Consider the set cover problem to find the smallest cover of the set X from the collection S_1, S_2, \dots, S_m of subsets of X . Pose this problem as a linear-programming problem.

Let X be a finite set of size n . We are given a family S_1, S_2, \dots, S_m of subsets of X with the property that $\cup_{i=1}^m S_i = X$. The MIN_SET_COVER problem deals with the computation of a smallest sub-collection C of the given family, which continues to cover X (that is, the union of the sets in C is X). We introduce a variable x_i for each S_i with the interpretation that

$$x_i = \begin{cases} 1 & \text{if } S_i \text{ is included in the cover } C, \\ 0 & \text{otherwise.} \end{cases}$$

The objective is to

$$\text{minimize } \sum_{i=1}^m x_i.$$

In order to insure that the sub-collection C covers X , take any element $a \in X$. Denote $\text{Idx}(a) = \{i \in \{1, 2, \dots, m\} \mid a \in S_i\}$. In order that a is covered by C , we require C to contain S_i for at least one $i \in \text{Idx}(a)$. Thus, we have the following constraints:

$$\sum_{i \in \text{Idx}(a)} x_i \geq 1 \quad \text{for each } a \in X.$$

7. Let f denote the maximum frequency of an element of X in the subsets U_j . Relax the LP of the last question. Propose a rounding criterion to arrive at an f -approximation algorithm.

We use the ILP formulation of the MIN_SET_COVER problem discussed in Section 26.3.2. The relaxation step replaces the constraints $x_i \in \{0, 1\}$ by $0 \leq x_i \leq 1$ for all $i = 1, 2, \dots, m$. Let $(x_1^*, x_2^*, \dots, x_m^*)$ be an optimal solution of the relaxed LP instance.

Rounding this solution is not done in the literal sense. For each $a \in X$, denote by f_a the frequency of a , that is, $f_a = |\text{Idx}(a)|$. Let $f = \max_{a \in X} f_a$ denote the maximum frequency. We take the rounded solution (x_1, x_2, \dots, x_m) as

$$x_i = \begin{cases} 0 & \text{if } x_i^* < 1/f \\ 1 & \text{if } x_i^* \geq 1/f \end{cases}$$

for all $i = 1, 2, \dots, m$.

To prove the feasibility of the rounded solution, we take any $a \in X$. The constraint $\sum_{i \in \text{Idx}(a)} x_i^* \geq 1$ must be satisfied by the relaxed optimal solution. If all the x_i^* values in the sum are $< 1/f$, we have $\sum_{i \in \text{Idx}(a)} x_i^* < f_a/f \leq 1$, a contradiction. So $x_i^* \geq 1/f$ for at least one $i \in \text{Idx}(a)$. For each such i , we have taken $x_i = 1$, that is, a is covered by the rounded solution.

The rounding method ensures that $x_i \leq f x_i^*$ for all $i = 1, 2, \dots, n$. Therefore the approximation ratio satisfies

$$\rho \leq \frac{\text{SUBOPT}}{\text{ROPT}} = \frac{\sum_{i=1}^n x_i}{\sum_{i=1}^n x_i^*} \leq \frac{\sum_{i=1}^n f x_i^*}{\sum_{i=1}^n x_i^*} = f,$$

that is, we have an f -approximation algorithm for the minimum set cover problem.

8. Propose an LP formulation of the SUBSET-SUM problem.

An LP-solver not only returns the optimum objective function but also WHERE it is achieved. So even if there is no objective, the feasibility of the constraints is an issue. So the LP formulation of SSP can be as follows.

$$\text{Minimize } 0 \text{ subject to } \sum_{i=1}^n x_i a_i = t.$$

Here, x_i is 1 if the i -th object is included in the subset; it is 0 otherwise.