BU CLA CS 530: Analysis of Algorithms
Spring 1999

Selected Solutions for Problem Set 14

Problem 1.

We show that MAX-2-BIN-PACK is NP-hard by the reduction PARTITION$\leq_P$MAX-2-BIN-PACK. We can take PARTITION as a decision problem and MAX-2-BIN-PACK as an optimization problem (the latter as stated in the handout for Problem Set 14). Let $S = \{s_1, \ldots, s_n\}$ be an input instance of PARTITION. If the total weight $w = s_1 + \cdots + s_n$ is an odd integer, which can be checked out "quickly" (in polynomial time), then it is not possible to partition S into two subsets of equal weight. Assume therefore that the total weight w is an even integer. Define $B = w/2$. The desired input instance of MAX-2-BIN-PACK is the pair (S, B), which can be obtained from S "quickly", i.e. we have indeed a polynomial-time transformation from the input of PARTITION to the input of MAX-2-BIN-PACK.

There is a partition of S into two subsets of equal weight $w/2$ iff n objects in S (i.e. all of S) can be packed into two bins of capacity $B = w/2$. This establishes that PARTITION$\leq_P$MAX-2-BIN-PACK.

Problem 2.

Start by sorting the objects in S in increasing order, i.e., assume that $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n$. Let m' be the largest index such that the sum of the m' smallest elements does not exceed B, and let m'' be the largest index such that the sum of the m'' smallest elements does not exceed $2B$. We can pack the m' smallest elements in one bin, ignore the $(m'+1)$st smallest element, and pack the next $(m''-m'-1)$ smallest elements in the second bin. The result is to pack a total of $(m''-1)$ elements in two bins. Noting that an optimal solution cannot exceed m'' (why?), our approximation is at most one element away from it.

Problem 3.

Assume there is a polynomial-time approximation algorithm for VERTEX-COVER which has absolute error e. We show how this assumption contradicts P $\neq$ NP. For this, we reduce the exact (optimization) version of VERTEX-COVER to its approximation version.

Let $G = (V, E)$ be an input instance of VERTEX-COVER, with $|V| = n$ and $|E| = m$. Suppose that an optimal cover for G has k vertices. We construct a new graph $G' = (V', E')$ consisting of $(e+1)$ distinct copies of G, so that G' has $(e+1)n$ vertices and $(e+1)m$ edges. In particular, G' has an optimal cover with exactly $(e+1)k$ vertices, with k vertices in each of the $(e+1)$ copies of G.

We now run the approximation algorithm on G'. The result is a vertex cover for G' with at most $(e+1)k + e$ vertices. The vertices of this collection are distributed among the $(e+1)$ copies of G. Moreover, the vertices present in any copy of G must form a cover of G, so that at least k vertices of the collection must appear in any given copy. Thus, at least $(e+1)k$ of the vertices in the collection are accounted for, leaving e vertices. But these remaining e vertices must be distributed among $(e+1)$ copies, so that one copy did not receive any additional vertex. For that copy of G with exactly k vertices from the collection, the supposed approximation algorithm actually found an exact solution, i.e., an optimal solution. Identifying that copy of G with exactly k vertices is "cheap", i.e., it is merely a matter of scanning in polynomial time all $(e+1)$ copies and retaining that copy with the minimum number of vertices in its cover. Hence, the exact version of the problem is reduced in polynomial time to the absolute-error approximation version of the problem.

Problem 4.

As in Problem 37-1 in [CLR], page 983, we take an input instance of BIN-PACK to be a set $S = \{s_1, \dots, s_n\}$ where $n \geqslant 1$ and $0 < s_i < 1$ for every $1 \leqslant i \leqslant n$. The weight of S is simply $w(S) = s_1 + \cdots + s_n$. A *packing of size* $k \geqslant 1$ for S is a partition P of S into k subsets each of weight $\leqslant 1$, i.e.,

- $P = \{S_1, \dots, S_k\}$ such that $S = \bigcup_{1 \leqslant i \leqslant k} S_i$ and $S_i \cap S_j = \varnothing$ for all $1 \leqslant i < j \leqslant k$.
- $w(S_i) \leqslant 1$ for every $1 \leqslant i \leqslant k$.

Suppose there is a fully polynomial-time approximation scheme BIN-PACK-SCHEME which, on input S and ε where $0 < \varepsilon < 1$, runs in time $\mathcal{O}(p(n, 1/\varepsilon))$ for some polynomial p. BIN-PACK-SCHEME is guaranteed to find a packing of size k for S such that if k^* is the minimum size of any packing, then $(k - k^*)/k^* \leqslant \varepsilon$. Define a new algorithm **A** which takes an arbitrary instance S of BIN-PACK as input and which is defined by:

```
A(S)
1.    n ← number of objects in S
2.    return BIN-PACK-SCHEME(S, 1/(n + 1))
```

Clearly, algorithm **A** runs in time polynomial in n, because $1/\varepsilon = (n+1)$ in this case. Suppose $\mathbf{A}(S)$ returns a packing of size k for S. Because $(k-k^*)/k^* \leqslant 1/(n+1)$, we have $k \leqslant k^* + k^*/(n+1)$. But $k^*/(n+1) < 1$ because it is surely possible to pack the n objects into n separate bins. Hence, $k^* \leqslant k < k^* + 1$, which in turn implies that $k = k^*$ because k is an integer. Hence, $\mathbf{A}(S)$ finds an optimal packing for S in polynomial time. But this is impossible if P $\neq$ NP, because BIN-PACK is NP-hard.

Problem 5. (Problem 37-2, part ***a*** only, page 984, in [CLR])

The definition of $G^{(2)}$ given in [CLR] is appropriate, and you can use it to solve the problem. But, as suggested in the lecture, you can also use the following somewhat simpler formulation of $G^{(2)}$. The vertices of $G^{(2)}$ are:

$$V^{(2)} = \{\ (v, w) \mid v, w \in V\ \}$$

and the edges of $G^{(2)}$ are:

$$E^{(2)} = \{\ \{(v, w), (v', w')\} \mid \text{either } (v = v' \text{ and } \{w, w'\} \in E) \text{ or } \{v, v'\} \in E\ \}$$

This definition of $E^{(2)}$ is not equivalent to the one given in [CLR]: $E^{(2)}$ here is a subset of $E^{(2)}$ in the book. We purposely denote edges in E and $E^{(2)}$ by 2-element sets, in order to avoid any confusion with ordered pairs of vertices (the elements of $V^{(2)}$).

It suffices to show that G has a clique of size $p \geqslant 1$ iff $G^{(2)}$ has a clique of size p^2. Suppose G has a clique $C = \{v_1, v_2, \dots, v_p\}$ of size p. Then $G^{(2)}$ has the clique $C^{(2)} = \{(v, w) \mid v, w \in C\}$ of size p^2.

For the converse, suppose $G^{(2)}$ has the clique $C^{(2)}$ of size p^2, where $p \geqslant 1$, but G has no clique of size p. We will get a contradiction. Define the set D as:

$$D = \{ \ v \mid \text{there is } w \in V \text{ such that } (v, w) \in C^{(2)} \ \}$$

D is a clique in G. Hence, $|D| \leqslant p - 1$. Hence, one of the $|D|$ sets $F_v = \{w \mid (v, w) \in E\}$ for $v \in D$ must have p or more vertices. As each of these sets F_v is also a clique, we get a contradiction.

Problem 6.

Suppose there is a polynomial time approximation algorithm $\mathcal{A}_\varepsilon$ for the clique problem, with relative error ε. We first construct another polynomial-time approximation algorithm $\mathcal{A}_\delta$, based on the given $\mathcal{A}_\varepsilon$, but with relative error $\delta < \varepsilon$ for some δ to be defined.

$\mathcal{A}_\delta$ works as follows: Given input graph G, it constructs $G^{(2)}$ and then applies $\mathcal{A}_\varepsilon$ to it. From the clique $C^{(2)}$ of $G^{(2)}$, $\mathcal{A}_\delta$ selects the largest of the sets D and F_v for $v \in D$ (see the solution for the preceding problem). The resulting set C is a clique of G and satisfies:

$$|C| \geqslant \lceil (\sqrt{|C^{(2)}|}) \rceil$$

by the preceding problem. Let p be the size of the largest clique in G. Again, by the preceding problem, the largest clique in $G^{(2)}$ has p^2 vertices. As $\mathcal{A}_\varepsilon$ has relative error ε, we have

$$\frac{p^2 - |C^{(2)}|}{p^2} \leqslant \varepsilon \quad \text{or} \quad |C^{(2)}| \geqslant p^2(1 - \varepsilon)$$

So that $|C| \geqslant \sqrt{|C^{(2)}|} \geqslant p\sqrt{1-\varepsilon}$. Hence, $\mathcal{A}_\delta$ has relative error $1 - \sqrt{1-\varepsilon}$, which is less than ε. The desired $\delta = 1 - \sqrt{1-\varepsilon}$.

We next construct an approximation-algorithm $\mathcal{A}_\varepsilon$, with relative error $\varepsilon' < \varepsilon$, where ε' is arbitrarily small. For this, we repeat the above process recursively r times, to obtain a relative error

$$\varepsilon' = 1 - (1 - \varepsilon)^{(1/2^r)}$$

(To repeat the process once more means, in effect, to construct $G^{(4)}$, to repeat the process twice more means to construct $G^{(16)}$, etc.) We choose r large enough so that

$$2^r > \frac{\log(1-\varepsilon)}{\log(1-\varepsilon')}$$

to produce the desired approximation algorithm $\mathcal{A}_{\varepsilon'}$.

Assaf Kfoury
Created: 99.05.04