Advanced Algorithms Approximation Algorithms Teaching Block 1

TAs : Kheeran K. Naidu, Alex Carpenter

Problem sheet 4 (Solutions)

Please feel free to discuss these problems on the unit discussion board or directly with your colleagues. If you would like to have your answers marked, please either hand them in in person at a lecture or problems class. Submitted work will be marked as quickly as possible, ideally within one week of being handed in.

1. Consider the optimisation and decision versions of the Bin Packing.

   In the optimisation version we ask for assignment of all the packages to bins using the least possible number of bins such that the sum of sizes of all packages in a bin is at most 1. In the decision version we are given a positive integer K as part of the input and asked if there exists a packing of all the packages into at most K bins such that the sum of sizes of all packages in a bin is at most 1.

   (a) Show how to solve the decision version of bin packing using the optimisation version. How many calls to the optimisation version does your reduction require ?

   Solution. We only need to make one call to the optimisation version. Say that the value returned, the fewest bins that we can pack our items into, is B. If $B \leq K$, then we can certainly pack our items into at most K bins, and so return 'Yes'. Otherwise, we return 'No'.

   ✓

   (b) Show how to solve the optimisation version of bin packing using the decision version. How many calls to the decision version does your reduction require ?

   Solution. Let n be the number of items that we want to pack. First, we find out how many bins is optimal - let us call this value B. We know that $1 \leq B \leq n$, so we could determine the value of B using a simple binary search - if we get a 'Yes' consider the larger elements, and if we get a 'No' consider the lower ones. This takes at most $\log n + 1$ calls to the decision version.

   Consider the optimal bin packing of the items into B bins. Observe that if we merged the items in each bin into a single item with weight equal to the sum of weights, then we would have B items with optimal bin packing of size B. The goal now is to find these merged items using only the decision version of the problem, having already found B.

```
Input : A set $S$ of items, a fixed item $s \in S$ and the optimal bin
packing number $B$ of set $S$.
1. For each item $t \in S\backslash\{s\}$
2.     let $s'$ be a new item with the summed weight of $s$ and $t$
3.     if $(S\backslash\{s,t\}) \cup \{s'\}$ can be packed into $B$ bins [a]
4.         return $t$
5. return none
```

a. We assume that if the weight of an item is too large for a bin, the decision problem outputs 'No'.

FIGURE 1 – Subroutine for finding mergeable items using the decision problem.

In Figure 1, we give a subroutine that, given some item s of the set of items S with optimal bin packing number B, either finds another item $t \in S$ which belongs in the same bin as s or proves that s is in a bin all by itself. This is because only items in the same bin can be merged without increasing[1] the optimal bin packing number. We use this subroutine to merge the items of the same bin together, keeping track of which items have been merged. Once there are exactly B items remaining (which can be packed into B bins), we are done.

Each call to the subroutine makes at most $|S|-1$ calls to the decision version of the problem (the loop in line 3). Only if two items are merged, then the size of S decreases by 1, so the size of S must decrease exactly $n - B$ times. In the worst case (when $B = n$), we make n calls to the subroutine, none of which decrease the size of S, hence the total number of calls made to the decision version of the problem is at most $n(n-1)$.

Overall, to determine B and to find the optimal bin packing, we use at most

$$n(n-1) + \log n + 1$$

calls to the decision version of the problem.

✓

2. Minimum vertex cover is a problem in graph theory. Given a graph $G = (V, E)$, a vertex cover is a set of vertices such that each edge in the graph is incident to at least one of vertex in the set, thus all edges are "covered" by that set of vertices. The set of all vertices, V, is one valid example of vertex cover. The problem is NP-hard.
   (a) Give a simple greedy 2-approximation algorithm for the minimum vertex cover problem.

1. Observe that merging items will never decrease the bin packing number since the merged item has the same weight as the sum of the individual items.

(b) Show correctness for your algorithm.

Solution.

```
algQ2(G) :
    1. $C \leftarrow \emptyset$
    2. For each $(u, v) \in E(G)$
    3.         if $u \notin C$ and $v \notin C$
    4.                 $C \leftarrow C \cup \{u, v\}$
    5. return $C$
```

Correctness. Assume for a contradiction that C is not a vertex cover. At least one edge must not be covered by C, hence $\exists (u, v) \in E$ such that $u \notin C$ and $v \notin C$. However, when the edge was processed by the algorithm in line 3, it would have been added to C, a contradiction. To prove that C is a 2-approximation, let $M \subseteq E$ be the set of edges which are processed in line 4. Observe that for every edge (u, v) in M, no other edge incident to u or v is added to M [2], by line 3. Therefore, any optimal minimum vertex cover C^* must include at least one endpoint of each edge in M, $|M| \leq |C^*|$. Since C is simply the endpoints of the edges in M, $|C| = 2|M| \leq 2|C^*|$ implying that C is a 2-approximation.

✓

3. Prove there is no FPTAS for minimum vertex cover unless P = NP.

Solution. Suppose we have an FPTAS for minimum vertex cover. Then for all $\epsilon > 0$, if *Opt* is a minimum vertex cover, we can guarantee a vertex cover C such that $|C| \leq (1 + \epsilon)|Opt|$. Importantly, we can produce such a C in time polynomial in both n and $\frac{1}{\epsilon}$. So the idea here is that if we set ϵ to be small enough, we could get a polynomial time algorithm that guarantees that $|C| = |Opt|$.

Let $\epsilon = \frac{1}{2n}$, where n is the number of vertices. Then $|C| \leq (1 + \epsilon)|Opt|$ gives us that $|C| \leq |Opt| + \frac{|Opt|}{2n}$. Since the size of a vertex cover can be no more than n, $\frac{|Opt|}{n} \leq 1$, and thus $|Opt| + \frac{|Opt|}{2n} \leq |Opt| + \frac{1}{2}$. So we have that $|C| \leq |Opt| + \frac{1}{2}$.

But the size of a vertex cover can only take integer values, so the above tells us that the only value that $|C|$ could take is exactly $|Opt|$ - that is, we have an exact solution for minimum vertex cover, an NP-hard problem, that can be computed in polynomial time (since $\frac{1}{\epsilon}$ is also polynomial in n). This is what it means for P = NP to hold, so we are done.

✓

2. i.e. M is a (maximal) matching.