CSCE 629-601 Homework 12
7th December, 2016

Name: Adil Hamid Malla **UIN:** 425008306

Problem 1. Textbook page 1111, Exercise 35-1.5.

Solution: No this doesn't imply that there is a polynomial time approximation algorithm with a constant approximation ratio for the clique problem. Let us prove this by the contradiction. As we know that the Vertex-Cover Approximation Algorithm ensures that the solution is at maximum the 2-Approximation Algorithm. We will assume that using the Vertex Cover Approximation Algorithm we can get the constant c-Approximation Algorithm for Maximum-Size Clique. Since we know according the theorem, that the relation between the Vertex - Cover and the Maximum Clique is the complement of the graph and also the the relation between the optimal solution of the graph is given by the relation that if the Maximum Clique of the graph is k, then the minimum vertex cover is given by the formula V -k, where V is the number of the vertices of the graph.
W know that we have a 2-approximation algorithm, call it A, for the optimization version of the vertex cover problem. We now propose a scheme to use A to create an approximation algorithm for clique (that is, given input G, the goal is to find a set of vertices that form a maximum-sized clique). First we compute G'. Next run A, the vertex cover approximation algorithm, on G" to obtain the approximate vertex cover $V' \in V$. Then output $V - V'$ (i.e. all vertices not in V') as the clique approximation.

Now lets consider that we have a Graph $G = (V, E)$. Let C^* be the size of the maximum Clique in the Graph G. And, let C be the size of the clique output using the above approximation algorithm. We now prove that there exists some input of G for which the ratio between the approximation and optimal algorithmic answer is greater than the constant value $c = 2$. This will be the contradiction to our assumption that the Clique can be solved in constant approximation ratio. Let the graph G has a maximum clique of V' of size $\frac{K}{2}$ where K is the number of the vertices in G. According the theorem 34.12, the vertices $V - V'$ form a minimum cover for the Graph G'. Hence for the graph G, given the minimum vertex cover in G' has size $K - \frac{K}{2}$ that is equal to $\frac{K}{2}$. When running the vertex cover approximation algorithm we are guaranteed to get the vertex cover of size at most twice that of the optimal vertex cover. Thus, in this case we are given the guarantee that $V' \leq 2(\frac{K}{2}) = K$. Hence, the final output $V - V'$ could contain no vertices i.e a clique of size 0. So the fraction $\frac{C^*}{C} = \frac{\frac{K}{2}}{0} = \infty$, which is greater than c =2, in that matter it is greater than any constant. So we have a contradiction, that we can have the solution to the Maximum Clique problem in constant-approximation. Moreover, the approximation algorithm will increase exponentially with each input of the graph G as the one unit difference in the minimum vertex cover problem, as minimum vertex cover problem results in the twice the number of the vertex in optimal solution.

Problem 2. Textbook page 1134, Exercise 35-1

a. *Prove that the problem of determining the minimum number of bins required is NP-hard.*
Solution:
We can show a "yes" certificate for the Bin-Packing (BP) problem. The certificate is a set of partitions of the objects, such that the number of partitions is less than K. We can verify in polynomial time if the sum of each partition is less than 1 and also if the the sets are disjoints and cover all the objects. Hence the BP $\in$ NP. To prove NP-Hard, we show a polynomial reduction from Subset-Partition(SP) problem(defined below) to Bin-Packing(BP) problem. Again, we can show a mapping from Subset-Sum(SS) problem to Subset-Partition problem. Since we know the SubSet-Sum problem is NP-Complete, the other problems are NP-Complete as well.

We formally define below the three decision problems mentioned above.

Bin-packing (BP) decision problem: Given a finite set S of objects, each with a size $s_i \in (0, 1)$, and a bound t, is it possible to pack all the objects (without breaking them) into at most K bins, where each bin has size 1?

Subset-sum (SS) decision problem: Given a finite set S of natural numbers and a target natural number t, is there a subset of S whose elements sum to t?

Set-partition (SP) decision problem: Given a finite set S of natural numbers, is there a partition of S into two mutually exclusive and totally exhaustive subsets such that the sum of elements is same in both the sub-sets?

SS $\leq_p$ *SP :* Let $s_1, \ldots, s_n, t$ be any SS input. We can transform it into an SP input $s_1, \ldots, s_n, 2S - t, S + t$, where $S = \sum_{i=1}^{n} s_i$. Note that for the SP problem, we are looking for a subset whose elements sum to $2S$ (since the entire input sizes sum to $4S$).

Correctness:
If a subset of $s_1, ..., s_n$ that sums to t exists, then it along with $2S - t$ sums to $2S$, and thus the SP input has a partition. Also, the second partition sum is $4S - 2S = 2S$

If a partition of the SP input exists, then the elements $2S - t$ and $S + t$ must be on different sides of the partition. In this case, the other elements on the side of partition containing $2S - t$ sum to t, and these elements show that the SS input has a subset summing to t.

SP $\leq_p$ *BP :* Let $s_1, ..., s_n$ be any SP input. We can transform it into a BP input with sizes $s_1/(S/2), ..., s_n/(S/2)$, with number of bins equal to 2.

Correctness:
If a partition of the SP input exists, then each side of the partition sums to $S/2$, so in the BP input, the corresponding objects' sizes sum to $(S/2)/(S/2) = 1$. Hence 2 bins suffice for the BP input.

If 2 bins are sufficient for the BP input, then the sizes of objects in each bin sum to at most 1. Thus the corresponding elements in the SP input sum to at most S/2. Since S is the sum of all the elements in the SP input, there must be two bins and each one must have sizes that sum to exactly 1. Thus the corresponding elements in the SP input sum to exactly S/2, and we have a partition.

b. *The first-fit heuristic takes each object in turn and places it into the first bin that can accommodate it. Let* $S = \sum_{i=i}^{n} s_i$.
Argue that the optimal number of bins required is atleast $\lceil S \rceil$
Solution: Consider a packing of n objects of sizes $s_1, s_2, \ldots s_n$ into K bins and $S = \sum_{i=i}^{n} s_i$. Let $u_1, u_2, \ldots u_K$ be the unused spaces in each bin and let $u = \sum_{i=i}^{n} u_i$. Hence,

$$\begin{aligned} K &= \frac{\text{Total space in bins}}{\text{size of a bin}} \\ &= \frac{\text{used space + unused space}}{1} \\ &= S + u \\ &\geq S \\ &\geq \lceil S \rceil \end{aligned}$$

c. *Argue that the first-fit heuristic leaves at most one bin less than half full*

Suppose two bins are less than half full. Before allocating the second bin, the first bin could have been filled with the object(s) that was filled in second bin and which contradicts the first-fit heuristic strategy. Hence, two bins cannot be less than half full.

d. *Prove that the number of bins used by the first-fit heuristic is never more than* $\lceil S \rceil$
Solution: Consider a bin-packing solution given by the First-fit heuristics. We have

$$K = S + u = S + \sum_{i=1}^{K} u_i$$

W.L.O.G, we assume $K \geq 2$, since $b = 1$ is a trivial case. Now let us consider the following cases- (i) when all the bins are more than half full. (ii) Exactly one bin (lth bin) is less than half full. These are the only two possibilities based on part c.

case-1: If all bins are greater than half full, then $K = S + u \leq S + \frac{b}{2}$, hence $b \leq 2S \leq \lceil 2S \rceil$

case-2: If exactly one of the bins is less than half full, and let the l be its bin number. Based on the First-Fit strategy, $u_l + u_j < 1 \forall j \neq l$. Hence, we have

$$u = (\sum_{i \neq l,j} u_i) + u_j + u_l < (\sum_{i \neq l,j} \frac{1}{2}) + 1 = \frac{K}{2}$$

Therefore $K = S + u < S + \frac{K}{2} \implies K < 2S \leq \lceil 2S \rceil$. Hence proved.

e. *Prove an approximation ratio of 2 for the first-fit heuristic.*
Solution: Let K be the number of bins used by the First-Fit heuristic method and $K^\star$ be the optimal number of bins. From part b and d, we get $K \leq \lceil 2S \rceil$ and $K^\star \geq \lceil S \rceil$. Combining the two equations we get

$$K \leq \lceil 2S \rceil \leq 2\lceil S \rceil \leq 2K^\star$$

Hence, $\frac{K}{K^\star} \leq 2$ and hence the approximation ratio is 2.

f. *Give an efficient implementation of the first-fit heuristic, and analyze its running time.*

Pseudo Code:

Algorithm 1 First-Fit$(n, s_1, s_2, \ldots s_n)$

```
for each object do
    scan the currently allocated bins until one with a free space is found
    if No bin has enough space then
        Allocate a new bin and put the object in that
    else
        Add the object in the bin found
    end if
end for
return bins and its objects
```

Time Complexity: In the worst case, each object goes in a new bin. So if there are n objects, then there are n bins. Iteration i of the for loop requires checking i-1 bins. So the total time is $O(n^2)$.

We can improve this by using a better data structure for the bins, based on the unused space. We can use a binary heap to manage this and the search complexity reduces to $O(n \log n)$ time.