COMP 5711 - Advanced Algorithms Written Assignment # 1

ZHAO, Xi
12256508
xzhaoca@connect.ust.hk

October 2, 2023

Fixed Parameter Algorithms (KT Ch 10)

Problem 1

Algorithm: Let $B_i = \{x_1, \ldots, x_c\}$ be any of the given sets in the instance. If $\mathcal{H}$ is a k-size hitting set, then at least one element in B_i must be in $\mathcal{H}$. If $x_i \in \mathcal{H}$, we can remove all sets B that contain x_i and then find a $(k-1)$-size hitting set for the remaining sets. We can consider all possible cases where different x_i belong to $\mathcal{H}$. Since there are a total of c different x_i, the k-size hitting set problem can be reduced to c $(k-1)$-size hitting set problems. Repeatedly, we solve the problem by reducing it to a 1-size hitting set problem. Such x_is constitute the final result. If there is a legal result found by this procedure, we return *yes*; otherwise, return *no*.

Correctness: If the algorithm finds a result, the result must be a legal k-size hitting set. If a k-size hitting set $\mathcal{H}$ exists, it can be found by this process because, for any set B_i, B_i has at least one element in $\mathcal{H}$.

Running Time: Let $T(k, m)$ be the running time of such a k-size hitting set problem with m instances; we have:

$$T(k, m) = cT(k-1, m-m_i) + m \log c \tag{1}$$

where m_i is the number of sets containing x_i and $m \log c$ is the time to check which sets contain x_i. So, $T(k, m) \le cT(k-1, m) + m \log c$ and $T(k, m) = O(c^k km \log c)$. Therefore, $f(c, k) = c^k k \log c$ and $p(m) = m$.

Problem 7

(a)

Let T be such a tree decomposition of G with tree width w, and let X be a bag in T. Then, the induced subgraph of G on X and its neighbors in T is a tree with at most $w+1$ vertices. We can color this tree with at most $w+1$ colors using a greedy algorithm. Start at the root of the tree and assign the first color to the vertices in X. Then, for each child of the root, assign the second color to the vertices in its bag that are not already colored. Continue this process for each level of the tree until all vertices are colored.

Now, we can extend this coloring to the entire graph G. For each bag X in T, assign to each vertex in X the minimum color that is not used by any of its neighbors in T. This is a valid coloring of G with at most $w+1$ colors, since each vertex has at most $w+1$ neighbors in T.

Therefore, the chromatic number of G is at most $w+1$.

(b)

We determine whether G is k-colorable for $k = 1, 2, \ldots, w+1$ by the dynamic programming. Let $(T, \{V_t : t \in T\})$ be a tree decomposition of G. For the subtree rooted at t, and every coloring X of V using the color set $(1, 2, ..., k)$, we use $f_t(X)$ to indicate whether there is a k-coloring of G_t by using X when restricted to V_t. We compute the values $g_t(x)$ when t is a leaf by simply trying all possible colorings of G_t. In general, suppose t has children $t_1, ..., t_d$, and we know the values of $g_{t_i}(X)$ for each choice of t_i and X. Then there is a coloring of G_t consistent with X on V if and only if there are colorings of the subgraphs $G_{t_1}, ..., G_{t_d}$ that are consistent with X on the parts of V_{t_i} that intersect with V_t. Thus, $g_t(X)$ is true if and only if there are colorings X_i of V_{t_i} such that $g_{t_i}(X_i)$ and X_i is the same as X when restricted to $V_t \cap V_{t_i}$. This requires us to maintain $k^{(w+1)} < (w+1)^{(w+1)}$ values for each piece of the tree decomposition.

Approximation Algorithms (KT Ch 11)

Problem 7

(a)

We process the customers in an arbitrary order. At any given point in time, let V_j denote the total value of all customers who have been shown ad j. As we see each new customer, we show him the ad for which V_j is the smallest. Let s' denote the spread of this algorithm. Suppose that ad j is the one achieving the minimal spread. Let i be any other ad and c be the last customer to be shown ad i. Before c was shown ad i, the value of V_i was at most V_j (by the definition of our greedy algorithm), and so $V_i \le V_j + v_c < V + \bar{v}/(2m)$, then

$$\bar{v} = \sum_{k=1}^{m} V_k < V_j + (m-1)(V_j + \bar{v}/(2m)).$$

Hence, $V_j > \bar{v}/(2m)$ by solving the inequality.

(b)

Assume that $n = 9$ and $m = 2$, 8 customers have the same value $v_i = l$ and a customer c_j has the value 2. Then our greedy algorithm will produce a spread of 4 when c_j appears at the last step, while the optimal spread is 5.

Problem 10

(a)

If $v \notin S$, it must have been deleted in some iteration by the selection of a node v'. This node v' must be a neighbor of v. Since v' rather than v is chosen, v' must have at least as much weight as v.

(b)

Consider any other independent set T. For each node $v \in T$, if $v \in S$, then we charge w to itself. Otherwise, v is a neighbor of some node $v' \in S$ whose weight is at least as large as v. We charge v to v'. Now, if v is charged to itself, then no other node is charged to v, since S and T are independent sets. Otherwise, at most four neighboring nodes of no greater weight are charged to u. Either way, the total weight of all nodes charged to v is at most $4w(u)$. Since these charges account for the total weight of T, it follows that the total weight of nodes in T is at most four times the total weight of nodes in S.

Problem 11

We define a normalized weight $w'_i = \left\lfloor \frac{w_i}{\epsilon W/n} \right\rfloor \cdot n/\epsilon W$ for each item i. For those items whose weight exceeds W, they cannot be put in O and we remove them. Then, we find the solution $\mathcal{A}$ by the dynamic programming algorithm for the knapsack problem with small normalized weights. Since $w' \leq W' = n/\epsilon$, the total cost is $O(nW') = O(n^2/\epsilon)$. Next, we prove that the method can find the correct solution. We can find the subset of normalized weight at most W which achieves the greatest value V'. V' must be greater than V because the total sum of normalized weights in O is at most W. When we put all these items in our knapsack, each has a weight that may be up to $\epsilon W/n$ more than we thought due to rounding error, so we use a weight of at most $W + n(\epsilon W/n) = W(1+\epsilon)$.