Approximation Algorithms - Homework 2

Ilay Menahem, and Aaron Ross

January 4, 2026

Problem 1: Weighted Max-Cut Local Search

(a) Polynomial Convergence

Overview

Proof of Validity

Complexity Analysis

(b) Improved Bound with Specific Initialization

Algorithm Modification

Complexity Analysis

Problem 2: Hitting Set

Algorithm: Recursive Local Ratio

Approximation Analysis

Problem 3: Multiple Choice Maximum Coverage

Problem 4: Generalized Steiner Forest (GSF)

Let $G = (V, E)$ be a connected undirected graph and $T_1, T_2, \ldots, T_t \subseteq V$. Let $w : E \to \mathbb{R}_{\geq 0}$ be a non-negative weight function over the edges. A generalized Steiner forest is a subset $E' \subseteq E$, such that for every $u, v \in T_i$ $(1 \leq i \leq t)$ there is a path in E' connecting u and v. In the generalized Steiner forest problem the objective is to find a generalized Steiner forest of minimum weight. (a) Let $T = \bigcup_{i=1}^{t} T_i$. Assume $|T_i| \geq 2$ for every $1 \leq i \leq t$ and consider $w' : E \to \mathbb{R}$ defined as follows. For an edge (u, v), $w'(u, v) = |\{u, v\} \cap T|$. Show that any minimal generalized Steiner forest of G is a 2-approximation with respect to the weight function w'. Note: a generalized Steiner forest E' of G is minimal if there is no $F \subsetneq E'$ which is also a generalized Steiner forest of G. (b) Suggest a 2-approximation algorithm for the generalized Steiner forest problem using the local ratio technique.

(a) Minimal GSF is a 2-approximation for w'

Claim: Any minimal generalized Steiner forest E' of G is a 2-approximation with respect to the weight function w'.

Lemma 1. *For any T and E the following equality holds:*

$$w'(E) = \sum_{v \in T} \deg_E(v)$$

Proof.

$$w'(E) = \sum_{(u,v) \in E} w'(u, v) = \sum_{(u,v) \in E} |\{u, v\} \cap T| = \sum_{(u,v) \in E} (\mathbf{1}_{u \in T} + \mathbf{1}_{v \in T})$$

$$= \sum_{v \in T} |\{(u, v) \in E\}| = \sum_{v \in T} \deg_E(v)$$

□

Proof. **Lower Bound on OPT:** Consider a optimal solution E^*. Since $\forall 1 \leq i \leq t, |T_i| \geq 2$, we know that any $v \in T$ must be connected to at least one other vertex in T_i via some edge in E^*. Thus, each vertex in T contributes at least 1 to the total weight of E^*. Therefore, we have:

$$OPT = w'(E^*) = \sum_{v \in T} \deg_{E^*}(v) \geq \sum_{v \in T} 1 = |T|$$

Upper Bound on E': Let E' be a minimal generalized Steiner forest of G. Let $1 \leq k \leq t$ be the number of connected components in the subgraph $G' = (V, E')$, Since E' is minimal, removing any edge from any component would disconnect a path between some $u, v \in T_i$, Thus, each component is a tree. a connected subgraph of a tree is a tree, we will mark every connected component of vertices of T in the graph G' as $C_1, C_2, \ldots, C_k$. For a tree with m vertices, the sum of degrees is $2(m - 1)$.

$$w'(E') = \sum_{u \in T} \deg_{E'}(u) = \sum_{C_i} 2(|C_i| - 1) \leq 2(|T| - 1) - k \leq 2|T|$$

Ratio: Combining the bounds, we have:

$$w'(E') \leq 2|T| \leq 2OPT$$

Thus, E' is a 2-approximation with respect to the weight function w'. □

(b) Local Ratio Algorithm

we will use the local ratio technique to design a 2-approximation algorithm for the generalized Steiner forest problem, this will be done by recursively decomposing the weight functions into ones we can apply the result from part (a) on. in the end we will get a graph such that all edges connected to T have weight 0, and we can select any minimal generalized steiner forest of T.

Algorithm 1 2-Approximation for Generalized Steiner Forest using Local Ratio

```
1: function GSF($G = (V, E), T_1, T_2, \ldots, T_t, w$)
2:     if w is identically 0 then return $\emptyset$
3:     end if
4:     Define $\epsilon = \min_{(u,v) \in E | w(u,v) > 0, u \in T \vee v \in T} w(u, v)$
5:     Define
6: end function
```

Problem 5: Knapsack with Partition Constraints

Consider the following variant of the Knapsack problem. The input consists of n items $I = \{1, \dots, n\}$, where item $i \in I$ has a weight $w_i \in \mathbb{Z}^+$ and a value $v_i \in \mathbb{Z}^+$. The items are partitioned into disjoint sets $S_1, S_2, \dots, S_m$; that is, $\bigcup_{j \in [m]} S_j = I$ and $S_j \cap S_\ell = \emptyset$ for all $j \neq \ell \in [m]$. Moreover, each set S_j has a bound $k(j)$. Also, we are given a knapsack capacity $B \in \mathbb{Z}^+$. A solution for the problem is a subset of items $P \subseteq I$ such that $\sum_{i \in P} w_i \leq B$ and $|P \cap S_j| \leq k(j)$ for all $j \in [m]$. The value of the solution P is $\sum_{i \in P} v_i$. The objective is to find a solution P of maximal value. Obtain an FPTAS for the problem.

Overview

we will maintain a *domination list* (a list of nondominated $(\text{weight}, \text{value})$ pairs). A pair (w, v) *dominates* (w', v') if $w \leq w'$ and $v \geq v'$. Dominated pairs can be discarded without affecting optimality.

For the FPTAS, we will apply the standard scaling technique (as in the usual knapsack FPTAS) and run the same domination-list DP on the scaled instance.

Algorithm Description

Domination-list primitives.

- $\textsc{Shift}(L, \Delta w, \Delta v) = \{(w + \Delta w, v + \Delta v) : (w, v) \in L\}$.
- $\textsc{Prune}(L)$ removes all dominated pairs from L (and may also remove pairs with $w > B$). One implementation: sort pairs by increasing weight; sweep, keeping a pair only if its value is strictly larger than the maximum value seen so far. Its complexity is $O(|L| \log |L|)$ due to sorting.
- $\textsc{Convolve}(L_1, L_2)$ computes all pairwise sums of pairs from L_1 and L_2:

  $$\textsc{Convolve}(L_1, L_2) = \{(w_1 + w_2, v_1 + v_2) : (w_1, v_1) \in L_1, (w_2, v_2) \in L_2\}$$

  . Its complexity is $O(|L_1| \cdot |L_2|)$.

Algorithm 2 Knapsack with Partition Constraints (Domination Lists)

```
1: function KNAPSACK-PARTITION($I, S_1, S_2, \dots, S_m, k, B$)
2:     $L \leftarrow \{(0,0)\}$                 ▷ global nondominated set of feasible (weight,value) pairs
3:     for each partition $S_j$ do
4:             ▷ Compute nondominated options using items from $S_j$ with at most $k(j)$ picks
5:         $L^{(0)} \leftarrow \{(0,0)\}$
6:         for $c = 1$ to $k(j)$ do $L^{(c)} \leftarrow \emptyset$
7:         end for
8:         for each item $i \in S_j$ do
9:             for $c$ from $k(j)$ down to 1 do
10:                $L^{(c)} \leftarrow \text{PRUNE}\Big(L^{(c)} \cup \text{SHIFT}(L^{(c-1)}, w_i, v_i)\Big)$
11:            end for
12:        end for
13:                                  ▷ Combine the chosen items from $S_j$ with the global list
14:        $L_{\text{next}} \leftarrow \emptyset$
15:        for $c = 0$ to $k(j)$ do
16:            $L_{\text{next}} \leftarrow \text{PRUNE}(L_{\text{next}} \cup \text{CONVOLVE}(L, L^{(c)}))$
17:        end for
18:        $L \leftarrow L_{\text{next}}$
19:    end for
20:    return $\max\{v : (w,v) \in L\}$
21: end function
```

Proof of Correctness

Without loss of generality, assume $\forall j,\ k(j) \leq |S_j|$ (otherwise replace $k(j)$ by $|S_j|$).

Lemma 2. *After processing partitions $S_1, \dots, S_j$, the list L contains exactly the nondominated pairs (w, v) such that there exists a feasible selection of items from the first j partitions of total weight w and total value v (respecting all bounds).*

Proof. **Base case ($j = 0$):** $L = \{(0, 0)\}$ is the unique feasible pair (select nothing), and it is nondominated.

Inductive step: Assume the claim holds after $S_1, \dots, S_{j-1}$. The lists $L^{(c)}$ are constructed with an explicit counter c, and PRUNE only removes dominated pairs, hence each $L^{(c)}$ represents exactly the nondominated achievable pairs using exactly c items from S_j. When convolving $L^{(c)}$ with L_{new}, we consider all ways to add c items from S_j to previously selected items (from partitions 1 to $j - 1$). The final pruning step ensures that only nondominated pairs remain in L_{new}. Thus, after processing S_j, L contains exactly the nondominated pairs corresponding to feasible selections from the first j partitions. □

Theorem 1. *The algorithm returns the optimal objective value for the knapsack with partition constraints problem.*

Proof. After processing all partitions, every feasible solution corresponds to some pair in the (unpruned) constructed set, and pruning preserves at least one representative of every attainable optimum value (for some weight). Thus $\max\{v : (w, v) \in L\}$ equals the optimal objective value under capacity B and the partition bounds. $\square$

Time Complexity

We will note $V = \sum_{i \in I} v_i$, and $k_{\max} = \max_{j \in [m]} k(j) \leq n$. The outer loop runs m times (once per partition).

Inside, we have two main parts:

- Updating $L^{(c)}$ for each item in S_j: we do $|S_j| \leq n$ iterations, each involving at most $k_{\max}$ shifts and prunes. Each prune takes $O(|L^{(c)}| \log |L^{(c)}|)$ time, and $|L^{(c)}| \leq \min(B, V)$ (since weights are at most B and values at most V). Thus this part takes $O(n \cdot k_{\max} \cdot \min(B, V) \log(\min(B, V)))$ time per partition.
- Combining $L^{(c)}$ into L_{next}: we do at most $k_{\max}+1$ convolutions and prunes. Each convolution takes $O(|L| \cdot |L^{(c)}|) \leq O(\min(B, V)^2)$ time, and each prune takes $O(|L_{\text{next}}| \log |L_{\text{next}}|) \leq O(\min(B, V) \log(\min(B, V)))$ time. Thus this part takes $O(k_{\max} \cdot \min(B, V)^2)$ time per partition.

Thus the total complexity is $O(m \cdot n \cdot k_{\max} \cdot \min(B, V)^2) = O(m \cdot n^2 \cdot \min(B, V)^2)$, which is polynomial in n, m, and $\min(B, V)$.

FPTAS Construction

To enable the FPTAS, we will use the standard scaling technique. Let $V_{\max} = \max_{i \in I} v_i$, $K = \frac{\epsilon V_{\max}}{n}$.

Algorithm 3 FPTAS for Knapsack with Partition Constraints

```
1: function FPTAS-Knapsack-Partition($I, S_1, S_2, \ldots, S_m, k, B$)
2:     Scale values: for each $i \in I$, set $v'_i = \lfloor v_i / K \rfloor$
3:     return Knapsack-Partition($I, S_1, S_2, \ldots, S_m, k, B$)
4: end function
```

Proof. **Time Complexity:** The time complexity is $O\left(m \cdot n^2 \cdot \min(B, V')^2\right)$

$$\min(B, V') \leq V' = \sum_{i \in I} v'_i \leq \sum_{i \in I} \frac{v_i}{K} = \frac{\sum_{i \in I} v_i}{K} \leq \frac{n V_{\max}}{K} = \frac{n V_{\max}}{\epsilon V_{\max}/n} = \frac{n^2}{\epsilon}$$

Thus the time complexity is $O\left(m \cdot n^2 \cdot \left(\frac{n^2}{\epsilon}\right)^2\right) = O\left(\frac{m \cdot n^6}{\epsilon^2}\right)$, which is polynomial in n, m, and $\frac{1}{\epsilon}$. $\square$

Proof. **Approximation Ratio:** For the optimal set P^*, the difference between the true value and the scaled value is:

$$\sum_{i \in P^*} v_i - K \cdot \sum_{i \in P^*} v'_i < \sum_{i \in P^*} K = |P^*| K \leq nK = \epsilon V_{\max} \leq \epsilon \cdot OPT$$

Thus, the solution returned is at least $(1 - \epsilon) \cdot OPT$. $\square$