1 Set Multicover

Set multicover problem is a generalization of the set cover problem. As in the set cover problem, we are given a universe set \( U = \{ a_1, a_2, \ldots, a_n \} \), a collection of subsets \( S = \{ S_1, S_2, \ldots, S_k \} \), where \( S_i \subseteq U \) for \( i = 1, 2, \ldots, k \), and a cost function \( c : S \to \mathbb{Q}^+ \). Additionally, for each \( a \in U \), there is an integer \( \tau_a > 0 \), which specifies how many times \( a \) should be covered by the selected subsets. The goal is to cover all elements up to their coverage requirements at minimum total cost. It is allowed to pick the set \( S_i \in S \) any integer number \( k \geq 0 \) of times, and the cost of picking that set then becomes \( k \cdot c(S_i) \). Propose a polynomial-time approximation algorithm for the set multicover problem, which achieves approximation factor \( H_n \), where

\[
H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n},
\]

and prove its correctness and approximation factor. \textit{Hint:} for example, you may use dual fitting technique.

The proposed algorithm in Algorithm 1 is a generalisation of the basic set cover algorithm that computes the total weight of the element as a sum of the weight of the element each time it is picked. However, we record the price at each step when the element is picked independently using an iteration counter. In addition, the cost effectiveness function is defined as

\[
\alpha(S) = \frac{c(S)}{|S \setminus T|}
\]

where \( T \) is the set of elements that are fully covered.

In the following analysis we also need a simple observation that the value of the lowest \( \alpha(S_i) \) in each iteration is non-decreasing. This is clear as the set \( T \) is non-decreasing but the cost \( c(S) \) always remains the same. Hence also for each element \( a \in U \) the values \( price(a, i) \leq price(a, j) \) if \( i < j \). We denote the total price of an element by \( price(a) \) and therefore we have

\[
price(a) = \sum_{i=1}^{\tau_a} price(a, i) \leq \tau_a \cdot price(a, \tau_a).
\]

The correctness of this algorithm is obvious from the condition of the cycle. This condition finished if \( T = U \), but by definition \( T \) only contains elements that are fully covered \( T = \{ a : picked(a) = \tau_a \} \).
Algorithm 1 Algorithm set multicover

Input: Set $U$, $S$ and respective values $\tau_a$
Output: A counter for each element of $S$ that denotes how many times it is picked

1: $T = \emptyset$
2: $\forall a \in U$ fix $picked(a) = 0$
3: $\forall S_i \in S$ fix $count(S_i) = 0$
4: while $T \neq U$ do
5:   Find $S_i$ with lowest $\alpha(S_i)$
6:   for $a \in S_i \setminus T$ do
7:     fix $picked(a) = picked(a) + 1$
8:     fix $price(a, picked(a)) = \alpha(S_i)$
9:     if $picked(a) = \tau_a$ then
10:        $T = T \cup a$
11:   end if
12: end for
13: $count(S_i) = count(S_i) + 1$
14: end while
15: return $\forall S_i \in S$ output $count(S_i)$

1.1 Approximation factor

We are using the dual fitting technique to prove this approximation factor, therefore we need to give the linear program corresponding to the set multicover problem. Let $\tau = \max_{a \in U} \tau_a$. It is clear that any set should be covered at most $\tau$ times in the optimal solution, but we do not need this requirement because we are optimising the minimum value of the objective function that has non-negative coefficients for all variables. Therefore we just need the constraint that all sets should be picked non-negative amount of times.

We use a variable $x_S$ for each set $S$ that denotes how many times the set $S$ is picked into the multicover.

We get the following relaxed linear program as

\[
\begin{align*}
\min & \sum_{S_i \in S} c(S_i) \cdot x_{S_i} \\
\text{s. t.} & \forall a \in U \sum_{S_i : a \in S} x_{S_i} \geq \tau_a \\
& \forall S_i \in S x_{S_i} \geq 0
\end{align*}
\]

In addition, we can write out the dual problem for this as follows. We use a variable $y_a$ for each $a \in U$. 


\[
\max \sum_{a \in U} y_a \cdot \tau_a \\
\text{s. t.} \forall S_i \in S \sum_{a \in S_i} x_S \leq c(S) \\
\forall a \in U \ y_a \geq 0
\]

Let's denote the optimal integer solution of the set multicovery problem by \(\text{opt}\) and the optimal solution of the primal linear program as \(\text{opt}_f\) because it can have a fractional value. We have \(\text{opt}_f \leq \text{opt}\) because \(\text{opt}\) is also a feasible solution of the linear program. In addition, we know that any feasible solution of the dual problem is a lower bound on the value \(\text{opt}_f\).

We fix a feasible solution of the dual problem as \(y = (y_{a_1}, \ldots, y_{a_n})\) where

\[
y_a = \frac{\text{price}(a)}{H_n \cdot \tau_a}.
\]

### 1.1.1 \(y\) is a feasible solution of the dual problem

For this we have to show that the two constraints in the dual problem hold for our choice of \(y\). The condition \(y_a \geq 0\) is trivial as \(y_a\) is chosen as a fraction of two positive numbers. We have to show that for each \(S_i \in S\) we have \(\sum_{a \in S_i} y_a \leq c(S)\).

Consider a specific set \(S = \{a_1, \ldots, a_m\}\) and assume that the elements were fully covered by the algorithm in the same order \(a_1, \ldots, a_m\). Consider the iteration when Algorithm 1 fully covers an element \(a_i \in S\). This means that \(\text{picked}(a_i) = \tau_{a_i}\). At this stage \(S\) contains at least \(m - i + 1\) not fully covered elements. Hence, the given price

\[
\text{price}(a_i, \tau_{a_i}) \leq \frac{c(S)}{m - i + 1}.
\]

Therefore, by definition and earlier observation

\[
y_a = \frac{\text{price}(a)}{H_n \cdot \tau_a} = \sum_{i=1}^{\tau_a} \frac{\text{price}(a, i)}{H_n \cdot \tau_a} = \frac{\tau_a \cdot \text{price}(a, \tau_a)}{H_n \cdot \tau_a} = \frac{\text{price}(a, \tau_a)}{H_n} \leq \frac{c(S)}{H_n \cdot (m - i + 1)}
\]

Hence, the constraint becomes

\[
\sum_{a \in S} y_a \leq \frac{c(S)}{H_n} \sum_{i=1}^{m} \frac{1}{(m - i + 1)} = \frac{c(S)}{H_n} \sum_{i=1}^{m} \frac{1}{i} = \frac{c(S) \cdot H_m}{H_n} \leq c(S).
\]

The last inequality holds because \(n \geq m\) and hence \(H_n \geq H_m\).

Therefore, \(y\) is a feasible solution because for all sets \(S_i \in S\) the constraints of the dual linear program hold.
1.1.2 The approximation factor of the algorithm is $H_n$

Now we use the solution $y$ to give a lower bound to the values $\text{opt}_f$ and $\text{opt}$ and show that $H_n$ is the approximation factor of the proposed algorithm. For this we rely on the fact that the cost of the algorithm is defined by the cumulative prices of all elements.

$$\text{cost(Solution)} = \sum_{a \in U} \text{price}(a) = \sum_{a \in U} y_a \cdot H_n \cdot \tau_a = H_n \sum_{a \in U} y_a \tau_a \leq H_n \text{opt}_f \leq H_n \cdot \text{opt}$$

The first inequality results from the fact that $\sum_{a \in U} y_a \tau_a$ is the value of the objective function of the dual problem computed at $y$ which is a lower bound for the value of the optimal solution $\text{opt}_f$ of the primal problem.

Hence, we know that the cost for the solution the algorithm finds is bounded by $H_n$ factor of the optimal solution $\text{opt}$.

2 Minimum vertex cover in bipartite graph

Let $G(V, E)$ be a (finite) undirected graph, and let $c : V \rightarrow \mathbb{Q}^+$ be a cost function defined on the vertices in $V$. Consider the following LP program for solving the minimum weight vertex cover problem:

$$\begin{align*}
\text{minimize} & \quad \sum_{v \in V} c(v) \cdot x_v \\
\text{subject to} & \quad \forall e = \{u, v\} \in E : x_u + x_v \geq 1 \\
& \quad \forall v \in V : x_v \geq 0
\end{align*}$$

Here, for all $v \in V$, we defined indicator variables $x_v$, which are 1 if $v$ is in the cover, and 0 otherwise.

Show that if $G$ is a bipartite graph, then all extreme point solutions of this LP problem are integral. Conclude, that LP programming can be used for solving efficiently the minimum weight vertex cover problem in a bipartite graph.

Assume, by contradiction that for some bipartite graph $G(A \cup B, E)$ there exists a not fully integral extreme point solution $x = (x_1, \ldots, x_n)$. We want to show that actually non-integral $x$ can not be an extreme point of the feasible region for this problem. We know that the vertex cover problem in the general case is half-integral, meaning that each coordinate of the extreme point is either 0, 1 or $\frac{1}{2}$. Hence, we assume, that there exists at least one coordinate $x_v = \frac{1}{2}$ in $x$.

Let us define two new points $t = (t_1, \ldots, t_n)$ and $w = (w_1, \ldots, w_n)$ as follows. Without lessening the generality assume that the index of the coordinate denotes the vertex that the variable represents.

$$t_i = \begin{cases} 
  x_i & \text{if } x_i \text{ is integral} \\
  0 & \text{if } x_i = \frac{1}{2} \text{ and } i \in B \\
  1 & \text{if } x_i = \frac{1}{2} \text{ and } i \in A
\end{cases}$$

$$w_i = \begin{cases} 
  x_i & \text{if } x_i \text{ is integral} \\
  0 & \text{if } x_i = \frac{1}{2} \text{ and } i \in A \\
  1 & \text{if } x_i = \frac{1}{2} \text{ and } i \in B
\end{cases}$$
Note that in fact
\[ x = \frac{1}{2}t + \frac{1}{2}w , \]
hence if \( t \) and \( w \) are two feasible points where \( t \neq w \) then \( x \) can not be an extreme point because extreme points of a convex region can not be represented as such a combination of two feasible points. From the fact that the graph is bipartite we know that each edge has one endpoint in \( A \) and the other in \( B \). Based on that we can show that \( t \) and \( w \) are two feasible points. It is trivial to notice that for each \( i \in V \) we have \( t_i \geq 0 \) and \( w_i \geq 0 \). Hence, we have to show that the edge constraint also holds.

Consider an edge \( e = \{u, v\} \in E \) where \( u \in A \) and \( v \in B \). The constraint is that \( x_u + x_v \geq 1 \) It is clear that the constraint for this edge is satisfied if \( x_v, x_u \in \{0, 1\} \) because in this case we did not change these values for \( t \) and \( w \). Assume, that \( x_v = \frac{1}{2} \) (the analysis for \( x_u = \frac{1}{2} \) is analogous), then we have two possibilities for \( x_u \in \{\frac{1}{2}, 1\} \) because \( x_u = 0 \) would invalidate the constraint.

1. \( x_u = 1 \). In this case we have \( t_u = w_u = 1 \) and the constraints \( t_u + t_v \geq 1 \) and \( w_u + w_v \geq 1 \) hold trivially because \( t_v, w_v \geq 0 \).

2. \( x_u = \frac{1}{2} \). In this case we get \( t_u = 1, t_v = 0 \) and \( w_u = 0, w_v = 1 \). Hence we have \( t_u + t_v = w_u + w_v = 1 \) and the constraints holds.

Therefore we have constructed two feasible points \( t \) and \( w \) such that \( x = \frac{1}{2}t + \frac{1}{2}w \) which is a contradiction because we assumed that \( x \) is an extreme point.

In total, the previous argument showed that from the half-integrality of general vertex cover problem we can conclude that the vertex cover problem for bipartite graphs is integral. Therefore, for a bipartite graph, we can solve the relaxed linear program to find an integer solution which is then a solution to the corresponding integer linear program and therefore an exact solution to the bipartite vertex cover problem.

### 3 4-colouring and vertex cover

*It is known that any (finite, undirected) planar graph is 4-colorable in polynomial time. In other words, the vertices of any planar graph can be efficiently partitioned into 4 subsets, such that there is no edge between any two vertices in the same subset. Show how you can use an algorithm for 4-coloring of a planar graph to find a 3/2-approximation to vertex cover in the given (finite, undirected) planar graph. Hint: Use the half-integrality of the vertex cover.*

We know that the linear program solution for the vertex cover is half integral meaning that each coordinate of the solution is in \( \{0, \frac{1}{2}, 1\} \). For an integer solution we can only have 0 or 1, hence we need to use 4-coloring to change the linear programming solution for the values \( \frac{1}{2} \). Consider the Algorithm 3. We assume the weighted version where each vertex \( v \in V \) of the graph has a cost \( c(v) \), for a not weighted version we can set \( c(v) = 1 \) for each \( v \).

The idea of the algorithm is that we get the half-integral optimal solution and then we modify the fractional values based on the colouring. For the color with the most weight in the fractional coordinates we choose 0 and for the rest we change the fractional solution to be 1. We have to show the correctness and approximation factor of this algorithm.
Algorithm 2 Algorithm for vertex cover using 4-colouring

Input: Planar graph
Output: A minimum vertex cover

1: Formulate and solve a linear program
2: Use 4-colouring to color the graph
3: For each colour 1, 2, 3, 4 compute
   \[ \text{cost}(k) = \sum_{v \in V, x_v = \frac{1}{2}, \text{color of } v \text{ is } k} c(v) \]
4: Pick a color \( k \) with maximum \( \text{cost} \)
5: \( \forall v \in V, x_v = \frac{1}{2} \) set \( x_v = \begin{cases} 0 & \text{if } v \text{ has color } k \\ 1 & \text{otherwise} \end{cases} \)
6: \textbf{return} Values \( x_v \) for each \( v \in V \)

3.1 Correctness

We know that the linear program formulated for Exercise 2 is a correct linear program for vertex cover and that we find a correct half-integral solution for it using the solver for the linear program. We have to show that the modified values on the edges still satisfy the constraints. The constraint \( x_v \geq 0 \) is trivially true. We have to show that for each edge \( e = \{u, v\} \) we have \( x_u + x_v \geq 1 \). Clearly, if we have \( x_u = \frac{1}{2} \) then by the constraint we either have \( x_v = \frac{1}{2} \) or \( x_v = 1 \). In the latter case is trivial, because changing \( x_u \) does not invalidate the constraint if \( x_v = 1 \). Therefore, we need to consider the case where for some edge \( x_u = x_v = \frac{1}{2} \).

By definition we change the value of these variables to 0 or 1. If neither \( v \) nor \( u \) are of color \( k \) then we pick \( x_u = 1 \) and \( x_v = 1 \) and the constraint is trivially satisfied. However, assume that \( v \) is of color \( k \), then we change \( x_v = 0 \). By definition of 4-colouring \( u \) can not be of color \( k \) because all edges have endpoints of different color and edge \( e = \{u, v\} \) already has an edge of color \( k \). Therefore, we set \( x_u = 1 \) and the constraint is satisfied.

In total, the modified solution is still a feasible solution to the defined linear program and therefore a valid solution for the vertex cover.

3.2 Approximation factor is \( \frac{3}{2} \)

We know that there exists an optimal fractional solution with value \( \text{opt}_f \) that the linear program solver finds. In addition, we know that the integer optimal has a value \( \text{opt} \geq \text{opt}_f \).

We chose some color \( k \) with the total maximum weight of the non integral vertices. Denote the total cost of non-integral vertices as

\[ w = \sum_{v \in V, x_v = \frac{1}{2}} c(v) \leq 2\text{opt}_f . \]
The inequality holds because we know that $\frac{1}{2} w$ is already computed into the optimal solution as

$$
\text{opt}_f = \sum_{v \in V, x_v = 1} c(w) + \frac{1}{2} w .
$$

We know that the average weight for each color is $\frac{w}{4}$ and hence

$$
cost(k) \geq \frac{w}{4} .
$$

Analogously, the total cost of other colors has a value $\text{cost}(k) \leq \frac{3w}{4}$.

Now, we know that the value of the objective function changes because we decrease the value of elements of color $k$ and increase the others. In addition, the change is from $\frac{1}{2}$ to 0 or 1, therefore has a coefficient of $\frac{1}{2}$. Hence, we can write out the formula for the cost of our solution as follows.

$$
cost = \text{opt}_f + \frac{1}{2} \cdot \text{cost}(k) - \frac{1}{2} \cdot \text{cost}(k)
$$

Now, we can use the previous bounds to write it out as

$$
cost \leq \text{opt}_f + \frac{1}{2} \cdot \frac{3w}{4} - \frac{1}{2} \cdot \frac{w}{4} = \text{opt}_f + \frac{1}{2} \cdot \frac{w}{2} \leq \text{opt}_f + \frac{1}{2} \cdot \text{opt}_f = \frac{3}{2} \cdot \text{opt}_f \leq \frac{3}{2} \cdot \text{opt} .
$$

Hence, we have shown that our solution has the cost that is bounded by $\frac{3}{2}$ factor of the optimal solution to the problem.

### 4 Hitting set problem

**Hitting set problem** is defined as follows. Given a universe set $U = \{a_1, a_2, \ldots, a_n\}$, and a collection of subsets $T = \{T_1, T_2, \ldots, T_k\}$, where all $T_i \subseteq U$, find a subset $M \subseteq U$ of minimum size, such that $M$ hits every $T_i$, namely, $M \cap T_i \neq \emptyset$ for $i = 1, 2, \ldots, k$. Devise a primal-dual approximation algorithm for this problem. What is the approximation factor of your algorithm? Hint: take $\alpha = 1$. What is the value of $\beta$?

First of all consider the linear integer program corresponding to the hitting set problem. For each element $a \in U$ we define a variable $x_a$ and define that $x_a = 0$ if $a$ is not chosen into $M$ and $x_o = 1$ if $a$ is chosen.

$$
\begin{align*}
\min & \sum_{a \in U} x_a \\
\text{s. t.} & \forall T_i \in T \sum_{a \in T_i} x_a \geq 1 \\
& \forall a \in U \ x_a \in \{0, 1\}
\end{align*}
$$

The main constraint corresponds to the fact that at least one element from each $T_i \in T$ has to be chosen into $M$. As usually, we can relax this problem to have $0 \leq x_a \leq 1$ and also drop the
constraint $x_a \leq 1$ because we are considering a minimisation problem and setting $x_a > 1$ would not affect the constraints but would increase the value of the objective function. Hence we get the following program

$$\begin{align*}
\text{min} & \quad \sum_{a \in U} x_a \\
\text{s. t.} & \quad \forall T_i \in T \sum_{a \in T_i} x_a \geq 1 \\
& \quad \forall a \in U \ x_a \geq 0
\end{align*}$$

And we can also consider the dual problem

$$\begin{align*}
\text{max} & \quad \sum_{T_i \in T} y_{T_i} \\
\text{s. t.} & \quad \forall a \in U \sum_{a \in T} y_T \leq 1 \\
& \quad \forall T_i \in T \ y_{T_i} \geq 0
\end{align*}$$

Based on these formalisations we can describe an algorithm for this problem as defined in Algorithm 4.

**Algorithm 3** Algorithm for the hitting set problem
Input: Sets $U$ and $T$
Output: set $M$
1: Set $\overline{x} = 0$ and $\overline{y} = 0$
2: Set $M = \emptyset$
3: while All $T_i \in T$ are not covered do
4: \hspace{1em} Pick $T_i$ that is not covered ($T_i \cap M = \emptyset$)
5: \hspace{1em} Increase the variable $y_{T_i}$ until some constraint becomes tight
6: \hspace{1em} pick all variables $a$ corresponding to tight constraints
7: \hspace{1em} for each $a$ do
8: \hspace{2em} Set $x_a = 1$
9: \hspace{2em} Set $M = M \cup \{a\}$
10: \hspace{1em} end for
11: end while
12: return $M$

We have to show the approximation factor and correctness of this problem. Let's pick $\alpha = 1$ and $\beta = \max_{T_i \in T} |T_i|$ to be the maximum size of a set $T_i$. Then this algorithm is $\alpha \beta = \beta$ approximation. Moreover, the bound is tight. For the bound we have to show the slackness conditions.

### 4.1 Correctness

Final $M$ trivially satisfies the condition that for each $T_i \in T$ we have $M \cap T_i \neq \emptyset$ because this is the condition for the cycle in the algorithm to terminate.
In addition, $x = 0$ is not a feasible solution of the primal problem because it does not satisfy any of the subset constraints. Analogously, $y = 0$ is a feasible solution of the dual problem because it satisfies all the constraints.

4.2 Final $x$ and $y$ are feasible solutions

The final value of $y$ is trivially feasible because we only change the value of some $y_T$ within the bounds of the constraints.

The value of $x$ is feasible because all sets $T_i$ are covered, meaning that for each $T_i$ there exists $a \in T_i \cap M$ such that $x_a = 1$ which satisfies the constraint for the subsets.

4.3 Slackness conditions

4.3.1 Primal complementary slackness condition

The condition is that for each $a \in U$ we have either $x_a = 0$ or $\sum_{T: a \in T} y_T = 1$. This results from the fact that $\alpha = 1$.

In the beginning of the algorithm the condition is satisfied as $x_a = 0$. There is only one case when we modify $x_a$ and in this case we set $x_a = 1$ if the constraint for $a$ becomes tight, meaning that we increased some $y_T$ so that $\sum_{T: a \in T} y_T = 1$. Hence, this condition is satisfied.

4.3.2 Dual complementary slackness condition

The condition is that for each set $T_i \in T$ we either have $y_{T_i} = 0$ or $1 \leq \sum_{a \in T_i} x_a \leq \beta \cdot 1$. We have $\beta = \max_{T_i \in T} |T_i|$.

This constraint holds because $1 \leq \sum_{a \in T_i} x_a$ has to hold for a valid solution $x$ by the constraints in the primal problem. In addition $\sum_{a \in T_i} x_a \leq \beta$ always holds for $\beta = \max_{T_i \in T} |T_i|$ because we have

$$\sum_{a \in T_i} x_a \leq |T_i| \leq \beta .$$

4.4 Approximation factor

From the slackness conditions we can conclude that the approximation factor of this algorithm is $\alpha \cdot \beta = \max_{T_i \in T} |T_i|$.

4.5 The approximation factor is tight

The approximation factor of this algorithm is tight. For example, consider a case where the problem has only one set $T = \{T_1\}$. Despite the size of $T_1$ the optimal solution $M$ always only contains one element from $T_1$. Hence, $\text{opt} = 1$. However, this algorithm would choose the set $T_1$ and make all
the constraints tight, therefore it sets $M = T_1$ and gets a value $\text{cost} = |T_1|$ for any set $T_1$. Here obviously $\beta = |T_1|$. 