From data towards knowledge: Revealing signaling pathways through unifying knowledge mining and data mining of systematic perturbation data

Songjian Lu, Bo Jin, Ashley Cowart, and Xinghua Lu

1 Supplement

1.1 The maximum highly density bipartite subgraph problem is NP-hard

MAXIMUM HIGHLY DENSITY BIPARTITE SUBGRAPH problem: Given a bipartite graph $G = (V_1, V_2, E)$ and a real number r, decide if there exists a subgraph $G' = (V_1', V_2', E')$ such that any $u \in V_1'$, $\mathsf{degree}(u) \geq |V_2'|r$, any $v \in V_2'$, $\mathsf{degree}(v) \geq |V_1'|r$, and $|V_1'||V_2'|$ is maximized. (note: $\mathsf{degree}(u)$ is the degree of vertex u, i.e. how many vertices that are connected to u.)

Theorem 1.1 *The* MAXIMUM HIGHLY DENSITY BIPARTITE SUBGRAPH *problem is NP-hard.*

PROOF. We will prove that the MAXIMUM HIGHLY DENSITY BIPARTITE SUBGRAPH problem is NP-hard by reducing the k-CLIQUE problem, which is NP-hard, to it.

Given a graph $G_1 = (V, E_1))$, where $n = |V|$, and an positive integer $k < |V|$, we construct a bipartite graph $G_2 = (V_1, V_2, E_2)$ by the following way.

- For any $u_i \in V$, add $u_{i1}, u_{i2}, \ldots, u_{i(n+k)}$ to V_1 and $v_{i1}, v_{i2}, \ldots, v_{i(n+k)}$ to V_2. For convenience, we call u_{i1} and v_{i1} the *outer* vertices and $u_{i2}, \ldots, u_{i(n+k)}$, $v_{i2}, \ldots, v_{i(n+k)}$ the *inner* vertices.
- Connect u_{i1} to $v_{i1}, v_{i2}, \ldots, v_{i(n)}$.
- For $1 < t \leq n$, connect u_{it} to $v_{i1}, \ldots, v_{i(t-1)}, v_{i(t+1)}, \ldots, v_{i(n+k)}$.
- For $t > n$, connect u_{it} to $v_{i2}, \ldots, v_{i(n+k)}$.
- If $(u_i, u_j) \in E_1$, then connect u_{i1} to v_{j1} and v_{i1} to u_{j1}.

It is easy to know that the graph G_2 has $n(n+k)$ vertices in each side. For any i and $t > 1$, the degree of u_{it} (the inner vertex) in V_1 is $n+k-1$ and the degree of v_{it} (the inner vertex) in V_2 is $n+k-1$. For any i the degrees of u_{i1} (the outer vertex) in V_1 and v_{i1} (the outer vertex) in V_2 are the sum of n (connect to inner vertices) and the degree of u in V (connect to outer vertices).

Now we prove that the graph G_1 has a clique of size k if and only if the graph G_2 has a highly density component $G' = (V_1', V_2', E')$ such that any $u \in V_1'$, $\mathsf{degree}(u) \geq |V_2'|\frac{n+k-1}{k(n+k)}$, any $v \in V_2'$, $\mathsf{degree}(v) \geq |V_1'|\frac{n+k-1}{k(n+k)}$, i.e. $r = \frac{n+k-1}{k(n+k)}$, and $|V_1'||V_2'| = (k(n+k))^2$, which is maximized.

The proof of one direction is easy. Given graph G_1, if G_1 has a clique of size k, then the subgraph, which is made from the clique of size k, of G_2 is obvious a highly density component $G' = (V_1', V_2', E')$ such that any $u \in V_1'$, $\mathsf{degree}(u) \geq |V_2'|\frac{n+k-1}{k(n+k)} = k(n+k)\frac{n+k-1}{k(n+k)} = n+k-1$, any $v \in V_2'$, $\mathsf{degree}(v) \geq |V_1'|\frac{n+k-1}{k(n+k)} = k(n+k)\frac{n+k-1}{k(n+k)} = n+k-1$, i.e. $r = \frac{n+k-1}{k(n+k)}$, and $|V_1'||V_2'| = (k(n+k))^2$.

Next we prove that $|V_1'||V_2'| = (k(n+k))^2$ is maximized. If G_2 has a subgraph $G'' = (V_1'', V_2'', E'')$ such that any $u \in V_1''$, $\mathsf{degree}(u) \geq |V_2''|\frac{n+k-1}{k(n+k)}$, any $v \in V_2''$, $\mathsf{degree}(v) \geq |V_1'|\frac{n+k-1}{k(n+k)}'$, i.e. $r = \frac{n+k-1}{k(n+k)}$, and $|V_1''||V_2''| > (k(n+k))^2$, then $|V_1''| > (k(n+k))$ or $|V_2''| > (k(n+k))$.

Without loss of generality, let $|V_1''| > (k(n+k))$, then for any $v \in V_2''$, $\mathsf{degree}(v) > n+k-1$, i.e. $\mathsf{degree}(v) \geq n+k$. Hence, any $v \in V_2''$, v can only be outer vertex. Case 1: V_1'' does not have any inner vertices. Then for any $v \in V_2''$, $\mathsf{degree}(v)$ is at most $n-1$. This will cause contradiction. Case 2: V_1'' has inner vertices. As V_2'' does not have any inner vertices, any inner vertex $u \in V_1''$, $\mathsf{degree}(u)$ is at most 1. Then $|V_2''|\frac{n+k-1}{k(n+k)} \leq 1$. This will lead to $|V_2''| \leq \frac{k(n+k)}{n+k-1}$. Thus $|V_1''| \geq \frac{n+k-1}{k(n+k)}(k(n+k))^2 > n(n+k)$. This is also a contradiction. Therefore, $|V_1''||V_2''| \leq (k(n+k))^2$.

Now we prove the other direction. Suppose G_2 has has a highly density component $G' = (V_1', V_2', E')$ such that any $u \in V_1'$, $\mathsf{degree}(u) \geq |V_2'|\frac{n+k-1}{k(n+k)}$, any $v \in V_2'$, $\mathsf{degree}(v) \geq |V_1'|\frac{n+k-1}{k(n+k)}$, i.e. $r = \frac{n+k-1}{k(n+k)}$, and $|V_1'||V_2'| = (k(n+k))^2$, which is maximized, we show that the original graph G_1 has a clique of size k.

First, we claim that $|V_1'| = |V_2'| = k(n+k)$.

If $|V_1'| \neq |V_2'|$, without loss of generality, let $|V_1'| > |V_2'|$. Then $|V_1'| > k(n+k)$ and for any $v \in V_2'$, $\mathsf{degree}(v) > n+k-1$, i.e. $\mathsf{degree}(v) \geq n+k$. Hence, any $v \in V_2'$, v can only be outer vertex. Case 1: V_1' does not have any inner vertices. Then for any $v \in V_2'$, $\mathsf{degree}(v)$ is at most $n-1$. This will cause contradiction. Case 2: V_1' has inner vertices. As V_2' does not have any inner vertices, any inner vertex $u \in V_1'$, $\mathsf{degree}(u)$ is at most 1. Then $|V_2'|\frac{n+k-1}{k(n+k)} \leq 1$. This will lead to $|V_2'| \leq \frac{k(n+k)}{n+k-1}$. Thus $|V_1'| \geq \frac{n+k-1}{k(n+k)}(k(n+k))^2 > n(n+k)$. This is also a contradiction. Therefore, $|V_1'| = |V_2'| = k(n+k)$.

Second, we claim that for any $u_i \in V$, if G' includes any vertex u_{it} that is made from u_i, then G' includes all vertices , $u_{i1}, u_{i2}, \ldots, u_{i(n+k)}$ and v_{i1} to $v_{i1}, v_{i2}, \ldots, v_{i(n+k)}$, that are made from u_i.

By claim First, we know that any vertex u in V_1' or V_2', $\mathsf{degree}(u) \geq n+k-1$. If G' has vertex u_{it}, that is made from u_i in V, in V_1' and does not has $v_{it'}$, that is made from u_i in V, in V_2', then it is obvious that any inner vertices, that are made form u_i in V, cannot be in G'. Therefore u_{it} is an outer vertex and must be connected to $n+k-1$ outer vertices in V_2'. This is impossible. Similarly, we can prove that it is impossible that G' has vertex v_{it}, that is made from u_i in V, in V_2' and does not has $u_{it'}$, that is made from u_i in V, in V_1'. The second claim is proved.

From precious two claims, we know that G' can only be made from k vertices in V. Then to make the degree of any outer vertex in G' to be at least $n+k-1$, any outer vertex of G' in V_1' must connect to all k outer vertices in V_2' and any outer vertex of G' in V_2' must connect to all k outer vertices in V_1'. Hence, those k vertices in V must make a clique, i.e. the graph G_1 has a clique of size k.

Therefore we proved that the MAXIMUM HIGHLY DENSITY BIPARTITE SUBGRAPH problem is NP-hard. □

1.2 The greedy algorithm for the maximum highly density bipartite subgraph problem

Before we introduce the greedy algorithm, we have to introduce how to calculate the score of a subgraph $G' = (V_1', V_2', E')$. We let the score of the subgraph be $|V_1'||V_2'| - p(|V_1'||V_2'| - |E'|)$, where p is the penalty for the missing edges, i.e. to calculate the score, we subtract a penalty p for each missing edge from $|V_1'||V_2'|$, which is the total number of edges for the complete bipartite graph. In the project, we let $p = \frac{1}{1-r}$ and $r = 0.75$, where r is the connectivity ratio. We obtained many good expression modules by these settings of p and r.

The basic idea of our greedy algorithm is that first, choose 3 vertices from V_1 to the current V_1' (We will enumerate all subsets of size 3 in V_1). Then repeatedly add the current best vertex u of V_1 into the current V_1'. The best u means the score of the induces subgraph $G' = (V_1' \cup \{u\}, V_2', E')$ is the best, where $V_2' = \{v | v \in V_2$ and v connects to at least $r(|V_1'| + 1)$ vertices in $V_1' \cup \{u\}\}$. We stop adding new vertex if no vertex of V_1 can make the current subgraph with better score. The algorithm detail is followed.

1.3 The exact algorithm for the t-cover hitting set problem

The WEIGHTED t-COVER HITTING SET problem is formulated as follows:

Algorithm-1 HDSubgraph(G, r)
Input: A bipartite graph $G = (V_1, V_2, E)$ and a real number r.
Output: A highly density subgraph

1. $G_{sub} = \emptyset$; $Score_{best} = -1$;
2. **for each** subset S_1 of size 3 in V_1 **do**
3. $V_{remain} = V_1 - S_1$; $V_1'' = S_1$; $Status = 1$;
4. **while** $Status = 1$ **do**
5. $Score_{temp} = -1$; $G'_{sub} = \emptyset$; $Status = 0$;
6. **for each** $u \in V_{remain}$ **do**
7. $V_1' = V_1'' \cup \{u\}$; $V_2' = \{v | v \in V_2$ and v connects to at least $r|V_1'|$ vertices in $V_1'\}$;
8. Calculate the score of induced subgraph $G' = (V_1', V_2', E')$ and save the score to SC;
9. **if** $SC > Score_{temp}$ **then**
10. $Score_{temp} = SC$; $G'_{sub} = G'$;
11. **if** $Score_{best} < Score_{temp}$ **then**
12. $G_{sub} = G'_{sub}$; $Score_{best} = Score_{temp}$; $Status = 1$;
13. Assign V_1' of G_{sub} to V_1''; $V_{remain} = V_1 - V_1''$;
14. **return** G_{sub};

Figure 1: Greedy algorithm to find the highly density bipartite subgraph

> Given a universe set X of m elements, a weight function $w(x) : X \to R^+$, a family $\mathcal{T} = \{S_1, \ldots, S_n\}$ of subsets of X, and an integer t, find a subset $H \subseteq X$ of minimum weight such that every subset in $\mathcal{T}$ has at least t elements in H, where the weight of H is defined as $\sum_{x \in H} w(x)$. We denote an instance of the problem by $(X, \mathcal{T}, w, t)$ and call H the minimum t-cover hitting set.

Suppose H is a subset of X, we define $hit_j(H) = \{S | S \in \mathcal{T}$ and $|S \cap H| \geq j\}$, $hit^t(H) = (hit_1(H), hit_2(H), \ldots, hit_t(H))$, and $weight(H) = \sum_{u \in H} w(u)$. The algorithm is presented in Figure 2.

Before proving the correctness and time complexity of the algorithm, we give the basic idea of the algorithm. Each subset S of X has at most $t + 1$ coverage statuses for any partial solution H', i.e. S may have 0, 1, ..., $t - 1$, or at least t element(s) in H'. Our algorithm use dynamic programming technique, which we remember different combinations of coverage statuses of all subsets in the problem, i.e. if $\mathsf{hit}^t(H_1) = \mathsf{hit}^t(H_2)$, we only keep one copy of the partial solution. As the problem has n subsets of X, the total number of different combinations of the coverage statuses is at most $(t + 1)^n$.

We use another trick to save the time, where we sort all n subsets of X by their sizes from minimum to maximum, such as S_1', S_2', ..., S_n'. In searching the solution, we first make sure that S_1' has at least t elements in the partial solutions; them make sure that S_2' has at least t elements in the partial solutions, and so on. If there are many subsets whose sizes are bounded by d, then in the step of making S_k' have at least t elements in the partial solutions, there are at most 2^{dk} (suppose $2^{dk} < (t + 1)^n$) different combinations of coverage statuses (as first k subsets are related to at most dk elements in X). After that, since the coverage statuses of S_1', ..., S_k' are fixed to one value, the total number of different combinations of the coverage statuses is at most $(t + 1)^{n-k} < (t + 1)^n$, where $(t + 1)^{n-k}$ and 2^{dk} can be much better than $(t + 1)^n$.

Following two lemmas are needed in the proof the main theorem.

Lemma 1.2 *Let X be a set of n elements and $\mathcal{X} = \{(S_t, S_{t-1}, \ldots, S_1) | S_1 \subseteq S_2 \subseteq \ldots \subseteq S_t \subseteq X\}$. Then $|\mathcal{X}| \leq (t + 1)^n$.*

Proof. (of Lemma 1.2) We prove this lemma by mathematical induction on t. When $t = 1$, $\mathcal{X} = \{(S_1) | S_1 \subseteq X\}$. The lemma is obviously true.

Algorithm-2 Hitting-1$(X, \mathcal{T}, w, t)$
Input: An instance of the WEIGHTED t-COVER HITTING SET problem ($t \geq 1$)
Output: A minimum weight t-cover hitting set

1.1 Sort $\mathcal{T}$ into $\{S_1, S_2, \ldots, S_n\}$ such that $|S_i| \leq |S_j|$ for any $i < j$;
1.2 Compute $X_i = \cup_{j=1}^{i} S_j$, $k_i = |X_i|$ for all $1 \leq i \leq n$;
1.3 Sort X into $\{u_1, u_2, \ldots, u_m\}$ such that for any $1 < i \leq n$,
if $u_{i_1} \in X_{i-1}$, $u_{i_2} \in X_i - X_{i-1}$, then $i_1 < i_2$;
1.4 $H_m = X$;
1.5 $\mathcal{Q}_{old} = \{(hit^t(\emptyset), \emptyset)\}$; $\mathcal{Q}_{new} = \{(hit^t(\emptyset), \emptyset)\}$;
2 **for** $i = 1$ to m **do**
3 **for** each $P = (hit^t(H), H) \in \mathcal{Q}_{old}$ **do**
4.1 $P' = (hit^t(H \cup \{u_i\}), H \cup \{u_i\})$;
4.2 **if** $hit_t(H \cup \{u_i\}) = \mathcal{T}$ **then**
4.3 **if** $weight(H_m) > weight(H \cup \{u_i\})$ **then**
4.4 $H_m = H \cup \{u_i\}$;
4.5 **else**
4.6 **if** there is no $(hit^t(H'), H')$ in $\mathcal{Q}_{new}$ such that $hit^t(H') = hit^t(H \cup \{u_i\})$ **then**
4.7 $\mathcal{Q}_{new} = \mathcal{Q}_{new} \cup \{P'\}$; /* u_i hits some additional subsets */
4.8 **else** /* u_i does not hit additional subsets but may reduce the total weight */
4.9 find $(hit^t(H'), H')$ in $\mathcal{Q}_{new}$ such that $hit^t(H') = hit^t(H \cup \{u_i\})$;
4.10 **if** $weight(H') > weight(H \cup \{u_i\})$ **then**
4.11 replace $(hit^t(H'), H')$ with $(hit^t(H'), H \cup \{u_i\})$;
5.1 **if** $i = k_j$ for some $1 \leq j \leq n$ **then**
5.2 find j' which is the maximum of all j such that $i = k_j$;
5.3 remove any $P = (hit^t(H), H)$ from $\mathcal{Q}_{new}$ if $S_1, \ldots, S_{j'} \notin hit_t(H)$;
5.4 $\mathcal{Q}_{old} = \mathcal{Q}_{new}$;
6 **return** H_m;

Figure 2: Algorithm for solving the WEIGHTED t-COVER HITTING SET problem.

Suppose when $t = i$, the lemma is true, i.e. $|\{(S_i, S_{i-1}, \ldots, S_1) | S_1 \subseteq S_2 \subseteq \ldots \subseteq S_i \subseteq X\}| \leq (i+1)^n$. Then when $t = i + 1$, we have

$$\begin{aligned}
|\mathcal{X}| &= |\{(S_{i+1}, S_i, \ldots, S_1) | S_1 \subseteq S_2 \subseteq \ldots S_i \subseteq S_{i+1} \subseteq X\}| \\
&= \sum_{S_{i+1} \subseteq X} |\{(S_i, S_{i-1}, \ldots, S_1) | S_1 \subseteq S_2 \subseteq \ldots \subseteq S_i \subseteq S_{i+1}\}| \\
&= \sum_{|S_{i+1}|=0}^{n} \binom{n}{|S_{i+1}|} |\{(S_i, \ldots, S_1) | S_1 \subseteq S_2 \subseteq \ldots S_i \subseteq S_{i+1}\}| \\
&= \sum_{|S_{i+1}|=0}^{n} \binom{n}{|S_{i+1}|} (i+1)^{|S_{i+1}|} \qquad \text{/* by induction assumption */} \\
&= ((i+1)+1)^n.
\end{aligned}$$

Hence, when $t = i + 1$, the lemma is still true. Thus we proved the lemma. □

Lemma 1.3 *Let H_1, H_2, H' be three subsets of X such that $H_1 \cap H' = \emptyset$ and $H_2 \cap H' = \emptyset$. If $hit^t(H_1) = hit^t(H_2)$, then $hit^t(H_1 \cup H') = hit^t(H_2 \cup H')$.*

PROOF. (of Lemma 1.3) As $hit^t(H_1) = hit^t(H_2)$, $hit_i(H_1) = hit_i(H_2)$ for all $1 \leq i \leq t$. Hence for any

$1 \leq i \leq t$,

$$
\begin{aligned}
& hit_i(H_1 \cup H') \\
= \; & \{S | S \in \mathcal{T} \text{ and } |S \cap (H_1 \cup H')| \geq i\} \\
= \; & \{S | S \in \mathcal{T} \text{ and } |S \cap H_1| + |S \cap H'| \geq i\} \qquad /^* \text{ because } H_1 \cap H' = \emptyset \; ^*/ \\
= \; & \bigcup_{j=0}^{|H'|} \Big(\{S | S \in \mathcal{T} \text{ and } |S \cap H_1| \geq i-j\} \bigcap \{S | S \in \mathcal{T} \text{ and } |S \cap H'| = j\} \Big) \\
= \; & \bigcup_{j=0}^{|H'|} \Big(hit_{i-j}(H_1) \bigcap \{S | S \in \mathcal{T} \text{ and } |S \cap H'| = j\} \Big) \\
= \; & \bigcup_{j=0}^{|H'|} \Big(hit_{i-j}(H_2) \bigcap \{S | S \in \mathcal{T} \text{ and } |S \cap H'| = j\} \Big) \\
= \; & \bigcup_{j=0}^{|H'|} \Big(\{S | S \in \mathcal{T} \text{ and } |S \cap H_2| \geq i-j\} \bigcap \{S | S \in \mathcal{T} \text{ and } |S \cap H'| = j\} \Big) \\
= \; & \{S | S \in \mathcal{T} \text{ and } |S \cap H_2| + |S \cap H'| \geq i\} \\
= \; & \{S | S \in \mathcal{T} \text{ and } |S \cap (H_2 \cup H')| \geq i\} \qquad /^* \text{ because } H_2 \cap H' = \emptyset \; ^*/ \\
= \; & \mathrm{hit}_i(H_2 \cup H').
\end{aligned}
$$

Therefore, $hit^t(H_1 \cup H') = hit^t(H_2 \cup H')$ and the lemma is proved. □

Theorem 1.4 *The* WEIGHTED t-COVER HITTING SET *problem can be solved in* $O((t+1)^n mnt)$ *time and in* $O((t+1)^n nt)$ *space, where* m *is the size of the ground set and* n *is the number of subsets for the given instance. If furthermore the problem has at least* $\frac{n}{1+d/\log_2(t+1)}$ *subsets whose sizes are upper bounded by* d*, then the problem can be solved in* $O(((t+1)^{d/(d+\log_2(t+1))})^n mnt)$ *time and in* $O(((t+1)^{d/(d+\log_2(t+1))})^n nt)$ *space.*

PROOF. We first prove the *correctness* of the algorithm.

Given an instance $(X, \mathcal{T}, w, t)$ of the WEIGHTED t-COVER HITTING SET problem, let $X = \{u_1, u_2, \ldots, u_m\}$, where X is sorted as shown in **Algorithm-1** such that the order of elements in X is as $S_1, S_2 - X_1, \ldots, S_n - X_{n-1}$, where $X_j = \cup_{i=1}^{j} S_i$. Let $H^\ell = \{u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}\}$, whose elements are in the same order as in the sorted X (i.e. if $j_1 < j_2$ with respect to the index of H^ℓ, then $i_{j_1} < i_{j_2}$ with respect to the index of X), be the minimum t-cover hitting set. Let $H_j^\ell = \{u_{i_1}, u_{i_2}, \ldots, u_{i_j}\}$ for all $1 \leq j \leq \ell$.

To prove correctness, we **claim** that when the `for` loop in step 2 of **Algorithm-1** is at loop $i = i_j$ for all $1 \leq j \leq \ell$ (note that u_{i_j} is the jth element in H^ℓ and i_jth element in X), immediately before step 5.4 is executed, there exists a $P = (hit^t(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_j^\ell)$ and $weight(H) = weight(H_j^\ell)$. We prove this claim by mathematical induction on j.

Induction basis. In the case of $j = 1$ (i.e. $i = i_1$), **first** we prove that the program will generate the element we need. Since $(hit^t(\emptyset), \emptyset)$ is in $\mathcal{Q}_{old}$, $(hit^t(\{u_{i_1}\}), \{u_{i_1}\}) = (hit^t(H_1^\ell), H_1^\ell)$ will be obtained in step 4.1.

Then we prove that the element we need will be added into $\mathcal{Q}_{new}$. If no $(hit^t(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_1^\ell)$, then $(hit^t(H_1^\ell), H_1^\ell)$ is added into $\mathcal{Q}_{new}$ in step 4.7. If there exists a $(hit^t(H'), H')$ in $\mathcal{Q}_{new}$ such that $hit^t(H') = hit^t(H_1^\ell)$, it is obvious that $weight(H') \geq weight(H_1^\ell)$. Otherwise $H' \cup \{u_{i_2}, u_{i_3}, \ldots, u_{i_\ell}\}$ would be a t-cover hitting set with weight that is less than the weight of $\{u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}\} = H^\ell$, which causes contradiction that H^ℓ is the minimum t-cover hitting set (note:

here we use the lemma 1.3 and the fact that $H' \cap \{u_{i_2}, u_{i_3}, \ldots, u_{i_\ell}\} = \emptyset$ and $H_1^\ell \cap \{u_{i_2}, u_{i_3}, \ldots, u_{i_\ell}\} = \emptyset$). If $weight(H') > weight(H_1^\ell)$, then H' is replaced by H_1^ℓ in step 4.11.

Finally we prove that the element we need will not be removed in step 5.3. If $i = i_1 = k_j$ for some $1 \le j \le n$, let j' be the maximum among all those j such that $i = k_j$, then t can only be 1. Because if $t > 1$, note that $X_{j'} = \cup_{j=1}^{j'} S_j$ and $X_{j'} \cap \{u_{i_2}, u_{i_3}, \ldots, u_{i_\ell}\} = \emptyset$, all S_j for $j \le j'$ is only covered by one element in H^ℓ, then H^ℓ cannot be the minimum t-cover hitting set of the problem. If $t = 1$, it is obvious that S_j, for all $1 \le j \le j'$, must have one element in H_1^ℓ, i.e. $\{S_1, \ldots, S_{j'}\} \subset hit_t(H_1^\ell)$. Hence for any $P = (hit^t(H), H) \in \mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_1^\ell)$, P will not be removed in step 5.3.

From above three steps, we proved that there must be a $(hit^t(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_1^\ell)$ and $weight(H) = weight(H_1^\ell)$. Therefore, in the case of $j = 1$, the claim is true.

Induction step. Suppose that when $j < q < \ell$ the claim is true, i.e. there exists a $P = (hit^t(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_j^\ell)$ and $weight(H) = weight(H_j^\ell)$. Now we prove that when $j = q$, the claim is also true. By induction hypotheses, when $j = q-1$, there exists a $P = (hit^t(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_{q-1}^\ell)$ and $weight(H) = weight(H_{q-1}^\ell)$. In step 5.4 of the last loop, $\mathcal{Q}_{new}$ is assigned to $\mathcal{Q}_{old}$. Hence, when $j = q$, there exists a $P = (hit^t(H), H)$ in $\mathcal{Q}_{old}$ such that $hit^t(H) = hit(H_{q-1}^\ell)$ and $weight(H) = weight(H_{q-1}^\ell)$. Similar to the proof of case $j = 1$, we still use three steps to prove the claim in the case of $j = q$.

First we prove the the program will generate the element we need. As u_{i_q} is not in H (note that any element in X is visited only once and u_{i_q} is not used before), where $hit^t(H) = hit(H_{q-1}^\ell)$, by Lemma 1.3, $hit^t(H \cup \{u_{i_q}\}) = hit^t(H_{q-1}^\ell \cup \{u_{i_q}\}) = hit(H_q^\ell)$. Hence a $P = (hit^t(H \cup \{u_{i_q}\}), H \cup \{u_{i_q}\})$, where $hit^t(H \cup \{u_{i_q}\}) = hit(H_q^\ell)$ and $weight(H \cup \{u_{i_q}\}) = weight(H_q^\ell)$, will be generated in step 4.1.

Then we prove that the new element we need will be added into $\mathcal{Q}_{new}$. If no $(hit^t(H'), H')$ in $\mathcal{Q}_{new}$ such that $hit^t(H') = hit(H \cup \{u_{i_q}\})$, then $(hit^t(H \cup \{u_{i_q}\}), \{H \cup \{u_{i_q}\}\})$ is added into $\mathcal{Q}_{new}$ in step 4.7. If there exists a $(hit^t(H'), H')$ in $\mathcal{Q}_{new}$ such that $hit^t(H') = hit^t(H \cup \{u_{i_q}\})$, it is obvious that $weight(H') \ge weight(H \cup \{u_{i_q}\})$. Otherwise $H' \cup \{u_{i_{q+1}}, \ldots, u_{i_\ell}\}$ would be a t-cover hitting set with weight that is less than the weight of $\{u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}\} = H^\ell$, which causes contradiction that H^ℓ is the minimum t-cover hitting set. If $weight(H') > weight(H \cup \{u_{i_q}\})$, then H' is replaced by $H \cup \{u_{i_q}\}$ in step 4.11.

Finally we prove that the element we need will not be removed in step 5.3. If $i = i_q = k_j$ for some j, let j' be the maximum among all j such that $i = k_j$ for $1 \le j \le n$, then all $S_1, \ldots, S_{j'}$ must be in $hit_t(H_q^\ell)$, otherwise H^ℓ would not be a t-cover hitting set for the given instance (note that $H^\ell - H_{i_q}^\ell$ does not have any element in S_j for $1 \le j \le j'$). Hence, for any $P = (hit^t(H), H)$ such that $hit^t(H) = hit^t(H_q^\ell)$, P will not be removed from $\mathcal{Q}_{new}$ in step 5.3.

From above three steps, we proved that there must be a $(hit(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_q^\ell)$ and $weight(H) = weight(H_q^\ell)$. Therefore, in the case of $j = q$, the claim is also true.

Therefore, when $j = \ell$, we will find a $(hit(H), H)$ in $\mathcal{Q}_{new}$ such that $hit^t(H) = hit^t(H_\ell^\ell) = hit^t(H^\ell)$ and $weight(H) = weight(H^\ell)$, i.e. we will find the minimum t-cover hitting set.

Having proved the correctness of the algorithm, let us now consider the time complexity and space complexity of the algorithm. Step 2 loops $|X| = m$ times. Step 3 loops $|\mathcal{Q}_{old}|$ times. By Lemma 1.2, $|\mathcal{Q}_{old}| < (t+1)^n$ (note that $hit^t(H) = (hit_1(H), hit_2(H), \ldots, hit_t(H))$ and $hit_t(H) \subseteq \ldots \subseteq hit_2(H) \subseteq hit_1(H) \subseteq \mathcal{T}$). Steps 4.1 to 4.4 take $O(nt)$ time. Steps 4.6 to 4.11 can be finished in $O(\log_2(t+1)^n) = O(n \log_2(t+1))$ time if we use AVL tree to implement $\mathcal{Q}_{new}$ and $\mathcal{Q}_{old}$. Hence the total time complexity is $O((t+1)^n mnt)$.

In the case that $\mathcal{T}$ has at least $\frac{n}{1+d/\log_2(t+1)}$ subsets whose sizes are bounded from above by d, then when $i = \frac{dn}{1+d/\log_2(t+1)}$, both $\mathcal{Q}_{old}$ and $\mathcal{Q}_{new}$ have at most $2^{\frac{dn}{1+d/\log_2(t+1)}} = ((t+1)^{\frac{1}{1+\log_2(t+1)/d}})^n$ elements. Furthermore, when $i = \frac{dn}{1+d/\log_2(t+1)}$, any $P = (hit^t(H), H)$ in $\mathcal{Q}_{old}$ or in $\mathcal{Q}_{new}$, $hit_i(H)$, where $1 \le i \le t$, has $S_1, S_2, \ldots, S_{\frac{n}{1+d/\log_2(t+1)}}$. Hence, when $i > \frac{dn}{1+d/\log_2(t+1)}$, all elements in $\mathcal{Q}_{old}$ or in $\mathcal{Q}_{new}$ have at

most $(t+1)^{n-\frac{n}{1+d/\log_2(t+1)}}$ combinations of $hit^t(H)$, i.e. the size of $\mathcal{Q}_{old}$ or $\mathcal{Q}_{new}$ is always bounded from above by $(t+1)^{n-\frac{n}{1+d/\log_2(t+1)}} = ((t+1)^{\frac{1}{1+\log_2(t+1)/d}})^n$. Therefore, the total time complexity is $O(((t+1)^{\frac{1}{1+\log_2(t+1)/d}})^n mnt)$.

The space complexity is $O(|\mathcal{Q}_{old}|\cdot(\text{lengthes of elements in } \mathcal{Q}_{old})) = O(|\mathcal{Q}_{new}|)\cdot(\text{lengthes of elements in } \mathcal{Q}_{new}))$. The lengthes of elements in both $\mathcal{Q}_{old}$ and $\mathcal{Q}_{new}$ are bounded from above by $O(nt)$. Therefore, in the general case, the space complexity is $O((t+1)^n nt)$ and in the case sizes of many subsets in $\mathcal{T}$ are bounded from above by d, the space complexity is $O(((t+1)^{d/(d+\log_2(t+1))})^n nt)$. □