Iran University of Science &Technology
3TH INTERNATIONAL CONFERENCE ON COMBINATORICS, CRYPTOGRAPHY AND COMPUTATION

Proceedings of the 3rd International Conference on Combinatorics, Cryptography and Computation (I4C2018)

A special case of the weighted set cover problem

Javad Tayyebi[1] and Aboumoslem Mohammadi[2*]

1 Department of Industrial Engineering, Birjand University of Technology, Birjand, Iran. Email: javadtayyebi@birjandut.ac.ir

2 Department of Mathematics, Faculty of Sciences, Imam Ali University, Tehran, Iran. Email: a.mohammadi30@gmail.com

ABSTRACT

A weighted set-cover problem contains a finite set S and a collection Ω of its subsets. A nonnegative weight w_j is associated with each $U_j \in \Omega$. The problem is to find a subcollection of $\mho$ with minimum weight so that the union of its sets is equal to S. This problem is a classic NP-hard combinatorial problem. In this paper, we investigate a special case of the problem that can be solved in strongly polynomial time where any two sets either have empty intersection or overlap completely. It is shown how this case may be transformed to a minimum cut problem. An example is given to illustrate the method.

KEYWORDS: Set cover, minimum cut, polynomial-time algorithm.

1 INTRODUCTION

Suppose that $S = \{1, 2, \dots, m\}$ is a ground set and $\Omega = \{U_j\}_{j=1}^{n}$ is a subset of the power set of S such that the union of its sets is equal to S. A non-negative weight w_j is associated with each $U_j \in \Omega$. A subcollection $\{U_j : j \in J\}$ of Ω is called a cover if $\bigcup_{j\in J} U_j = S$. The weighted set cover problem consists of identifying a cover with minimum weight. This problem can be formulated as follows:

$$\begin{aligned} \min z = \sum_{j\in J} w_j, \\ s.t. \bigcup_{j\in J} U_j = S, \end{aligned} \qquad (1)$$

in which $J \subseteq \{1, 2, \dots, n\}$ is an index set to be determined. Any instance of problem (1) is denoted by the triple $(S; \Omega; w)$. This problem is equivalent to several combinatorial problems such as the hitting set problem, the hypergraph vertex cover problem, and the dominating set problem [3]. Problem (1) is one of the 21 problems whose NP-hardness was first established by Karp [4]. Thus, due to its inherent complexity, problem (1) cannot be solved efficiently in an exact way unless it is NP-hard. There exist four different remedies to this problem:

1) to apply implicit enumeration methods;
2) to design heuristic and metaheuristic methods;
3) to present polynomial-time approximation algorithms with guaranteed performance;
4) to propose polynomial-time exact algorithms for some special cases of the problem.

To the best of our knowledge, there is no work that considers the fourth remedy. In this work, we develop an algorithm for a special cases of problem (1) to obtain exact solutions in strongly polynomial time. The case is that any two sets of Ω either have no common elements or one is contained in the other. We call it 'the disjoint-set cover problem' and prove that it is converted into a minimum cut problem on a tree-like network. Then, we present a polynomial-time algorithm to find the minimum cut in the corresponding network without solving a maximum flow problem.

2 EFFICIENT ALGORITHM

In this section, we investigate problem (1) with the following extra assumption.

Assumption 2.1: *For each $i, j \in \{1, 2, \dots, n\}$, one of the three following conditions holds:*

- $U_i \subseteq U_j$;
- $U_j \subseteq U_i$;
- $U_i \cap U_j = \emptyset$.

Consider an instance (S, Ω, w) satisfying Assumption 2.1. We construct an auxiliary network $G(V, A, \overline{w})$ from (S, Ω, w) in the following manner. The network G has a vertex i for each $i \in S$,a vertex U_j for each $U_j \in \Omega$ and additionally, two vertices s and t. Then, the vertex set is $V = S \cup \Omega \cup \{s, t\}$. The arc set A is partitioned into the following subsets.

A1: If U_j has no superset in $\Omega - \{U_j\}$ then the arc (s, U_j) is added to G with $\overline{w}_{sU_j} = w_j$.

A2: If $U_i \subseteq U_j$ and there exists no $k \in \{1, \dots, n\} - \{i, j\}$ such that $U_i \subseteq U_k \subseteq U_j$, then the arc (U_j, U_i) is added to G with $\bar{w}_{U_j U_i} = w_i$.

A3: If $i \in U_j$ and there is no $k \in \{1, \dots, n\} - \{j\}$ such that $i \in U_k \subseteq U_j$, then the arc (U_j, i) is added to G with $\overline{w}_{U_j\, i} = +\infty$.

A4: For each $i \in S$, the arc (i, t) is added with $\overline{w}_{it} = +\infty$.

The network $G(V, A, \overline{w})$ has $n + m + 2$ vertices and $n + 2m$ arcs. Notice that one can simply construct G from an instance (S, Ω, w) in a time $O(n^2 m)$.
In the case where Ω contains no singleton set, the ground graph of the network G is the same as the Hasse diagram of the partially ordered set $(V, \subseteq)$ in which each $i \in S$ corresponds to the singleton set $\{i\}$ and two vertices s and t correspond to S and , respectively [2]. The following simple property is immediate.

Property 2.2: For any instance (S, Ω, w) satisfying Assumption 2.1 and the corresponding network $G(V, A, \overline{w})$,

- if $U_i \subseteq U_j$, then there exists at least one direct path from U_j to U_i in G;
- if $i \in U_j$, then there exists at least one direct path from U_j to i in G.

Lemma 2.2. The induced subgraph $G[V - (S \cup \{t\})]$ is an outtree rooted at s.

Proof. To establish the result, it is sufficient to prove the following claims.
Claim 1. If $U_i \subseteq U_j$, then there exists exactly one direct path from U_j to U_i in G.
Claim 2. For each $U_j \in \Omega$, there exists exactly one direct path from s to U_j in G.
Proof of Claim 1. The existence of such a path is the direct consequence of Property 2.2. By contradiction, assume that there exist two paths P and P' from U_j to U_i. Without loss of generality, we suppose that P and P' have no common vertices except U_j and U_i. Two distinct cases may occur: (I) at least one of the sets $P - P'$ and $P' - P$ is empty; (II) both the sets $P - P'$ and $P' - P$ are non-empty. In the first case, let $P - P' = \emptyset$. Then, $P = U_j - U_i$ and $P = U_j - U_k - \dots - U_i$. From path P', we have $U_i \subseteq U_k \subseteq U_j$. This contradicts the fact that arc $(U_j, U_i) \in A2$ exists on path P. In the second case, there exist two distinct indices $k, k' \in \{1, \dots, n\}$ such that $P = U_j - U_k - \dots - U_i$ and $P' = U_j - U_{k'} - \dots - U_i$. This implies that sets U_k and $U_{k'}$ do not satisfy Assumption 2.1 because $U_i \subseteq U_k \cap U_{k'}$.
Proof of Claim 2. For each $U_j \in \Omega$, consider superset U_k of U_j such that U_k contains no proper superset belonging to Ω. Based on Claim 1, there exists one path P from U_k to U_j. This path together with arc $(\mathrm{s}, \mathrm{U_k}) \in$ A1 provide a unique path from s to U_j.
Hereafter, we focus on st-cuts of G. Let us first recall the notion of an st-cut. For any $C \subseteq V$ with $s \in C$ and $t \notin C$, an st-cut $[C, \bar{C}]$ is the set of arcs whose tail belongs to C and whose head belongs to $\bar{C}$ in which $\bar{C}$ is the complement of C. The weight of an st-cut is the sum of the weights of its arcs, i.e.,$\sum_{(i,j) \in [C, \bar{C}]} \overline{w}_{ij}$.

Note that removing the arcs of an *st*-cut separates G into two subgraphs so that one contains s and the other contains t .

Theorem 2.4: Let (S, Ω, w) be an instance of problem (1) satisfying Assumption 2.1 and $G(V, A, \overline{w})$ be the corresponding network. A cover of (S, Ω, w) corresponds to an st-cut of G with finite weight and vice versa. Furthermore, their weights are the same.

Proof. Suppose that $\{Uj\}_{j \in J}$ is a cover of (S, Ω, w). From Lemma 2.3, we know that each vertex U_j has exactly one incoming arc (k_j, U_j) with weight $w_{k_j U_j} = w_j$ in which $k_j \in \Omega \cup \{s\}$. Consider the set $A' = \{(k_j, U_j): j \in J\}$. We show that the set A' is an st-cut of G. By contradiction, suppose that there exists a path P from s to t in the network $G(V; A - A'; \overline{w})$. Since incoming arcs of t emanate from the elements of S, it follows that $P = s - \cdots - i - t$ for some $i \in S$. Therefore, i is not included in any set of U_j , $j \in J$, which is a contradiction.

Based on Theorem 2.4, problem (1) satisfying Assumption 2.1 reduces to identifying a minimum *st*-cut in $G(V, A, \overline{w})$. One can solve the maximum flow problem defined on the network G to find its minimum *st*-cut [1]. However, it requires extra time for solving the maximum flow problem. Here, we present a strongly polynomial-time algorithm to identify a minimum *st*-cut directly.

We first mention some elementary notions in trees. From Lemma 2.3, we know that the graph $T = G[V - (S \cup \{t\})]$ is an out-tree rooted at s. We say that U_i is a 'predecessor' of U_j, denoted by $U_i = pr(U_j)$, if U_i is the next vertex on the unique path in T from U_j to s. If $U_i = pr(U_j)$, then we call U_j a 'successor' of U_i. The descendants of a vertex U_j, denoted by $des(U_j)$, consist of its successors, successors of its successors, and so on.

Lemma 2.5: If vertex U_j has an outgoing arc with infinite capacity, then no arc belonging to the induced graph $G[des(U_j)]$ is in the minimum cut of G.

Proof. Suppose that arc (U_j , i) exists and has infinite capacity. If an arc in $G[des(U_j)]$ belongs to an st-cut, then the st-cut also contains (U_j , i).Therefore, such an st-cut has infinite capacity and cannot be a minimum st-cut.

Using Lemma 2.5, we can prune the out-tree $G[V - (S \cup \{t\})]$. For every $U_j \in \Omega$, we remove the subtree $G[des(U_j)]$ from G if U_j has an outgoing arc with infinite capacity. We denote by $G(V, A, \overline{w})$ the network obtained after pruning. It is trivial that $G[V - (S \cup \{t\})]$ is also an out-tree rooted at s. Each leaf of $G[V - (S \cup \{t\})]$ has an outgoing arc with infinite capacity in G' and each infinite capacity arc (U_j, i) emanates from one leaf of the out-tree. Our proposed algorithm is stated as Algorithm 1.

Algorithm 1

```
Input: A network G(V, A,w) constructed from an instance of problem (1) satisfying Assumption 2.1.
Output: A minimum st-cut C of G.
Step 0: Set $G^0 = G$ and $C = \emptyset$;
Step 1: for $U_j \in \Omega$ do
            if $U_j$ has an outgoing arc with infinite capacity then
                remove $G^0[des(U_j)]$ from $G^0$;
            end
        end
Step 2: for $U_j \in \Omega$ do
            if $des(U_j) = \emptyset$ then
                $C = C \cup \{(pr(U_j), Uj)\}$;
            end
        end
Step 3: Apply the Breadth-First-Search algorithm to number the vertices belonging to $G'[V' - (S \cup \{t\})]$ starting $s$ as $\{s, U_1, \ldots, U_r\}$.
Step 4: for j = r to 1 do
            if $\overline{w}_{pr(U_j),U_j} \leq \sum_{(pr(U_k),U_k) \in C: U_k \in des(U_j)} \overline{w}_{(pr(U_k),U_k)}$ then
```

```
                for $(U_i, U_k) \in G'[des(U_j)]$ do
                        $C = C - \{(U_i, U_k)\}$;
                end
                $C = C \cup \{(pr(U_j), U_j)\}$;
        end
    end
```

In Step 1 of Algorithm 1, we prune the initial network to obtain the network $G(V, A, \overline{w})$. In Step 2, we add all leaf arcs of the out-tree $G'[V' - (S \cup \{t\})]$ to C. It is obvious that C is an st-cut with finite capacity. In Steps 3 and 4, we traverse vertices U_j from leafs toward s and check whether the unique incoming arc $(pr(U_j), U_j)$ can construct a new st-cut with capacity less than that of C if we replace it with the arcs in $G'\,[des(U_j)]$.

Lemma 2.6: Algorithm 1 finds a minimum st-cut in G in $O(n^2)$ time.

Proof. The running time of the algorithm is obviously $O(n^2)$. Here, we only prove its accuracy. Since C determined after Step 2 is an st-cut and remains an st-cut even after each iteration of Step 4, it follows that C obtained from the algorithm is also an st-cut. By contradiction, suppose that C' is a minimum st-cut whose capacity is less than that of C. Without loss of generality, we assume that C' is the nearest minimum st-cut to s, i.e. there is no other minimum st-cut C'' such that for each $(pr(U_j), U_j) \in C'$, C″ has no arc on the unique path from s to $pr(Uj)$. Necessarily, $C' - C \neq \emptyset$. Let $(pr(U_j), U_j) \in C' - C$. Consider the iteration of Step 4 in which U_j is checked. If

$$\overline{w}_{pr(U_j),U_j} > \sum_{(pr(U_k),U_k)\in C: U_k \in des(U_j)} \overline{w}_{pr(U_k),U_k},$$

then

$$C'' = (C' - \{(pr(U_j), U_j)\}) \cup \{(pr(U_j), U_j) \in C: U_k \in des(U_j)\}$$

is an st-cut with capacity less than that of C', which is a contradiction. If

$$\overline{w}_{pr(U_j),U_j} \leq \sum_{(pr(U_k),U_k)\in C: U_k \in des(U_j)} \overline{w}_{pr(U_k),U_k},$$

then $(pr(U_k), U_k)$ becomes a member of C in this iteration. On the other hand, we assumed tha $(pr(U_j), U_j) \in C' - C$. So it has to be removed from C in some iteration. Assume that it occurs in the iteration of Step 4 in which U_i is checked. Then

$$C'' = (C' - \{(pr(U_i), U_i)\}) \cup \{(pr(U_j), U_j) \in C: U_k \in des(U_i)\}$$

does not contain$(pr(U_j), U_j)$ and has a capacity less than or equal to C. It contradicts the assumption that C is the nearest minimum st-cut to s.

In summary, to solve problem (1) satisfying Assumption 2.1, we first construct the network G in a time $O(n^2 m)$. Then, we apply Algorithm 1 to find a minimum st-cut C of G in a time $O(n^2)$. The optimal solution of problem (1) is $\{U_j : (pr(U_j), U_j) \in C\}$, whose weight is equal to the capacity of the minimum st-cut problem. Thus, we have established the following result.

Theorem 2.7: Problem (1) satisfying Assumption 2.1 can be solved in a time $O(n^2 m)$.

Table 1. Data for Example 1.

j	U_j	w_j
1	{1, 3, 5, 7, 9}	9
2	{2, 4, 6, 8}	8
3	{1, 3, 5}	7
4	{7,9}	1
5	{2,4,6}	4
6	{8}	5
7	{1,5}	4
8	{3}	3
9	{2,4}	2
10	{2}	1

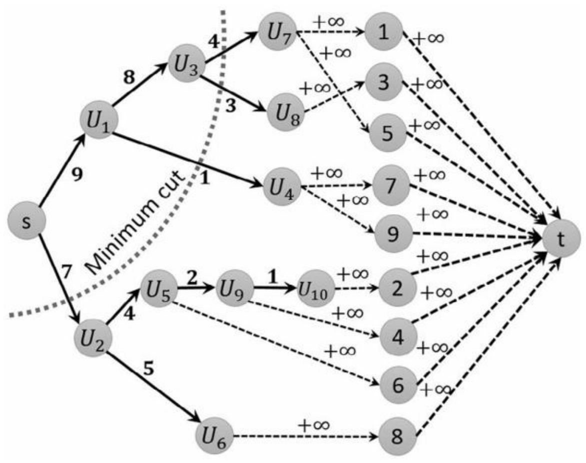

Figure 1. Network $G(V, A, \overline{w})$ for Example 1.

Example 1: Consider an instance of problem (1) in which $S = \{1, 2, \dots, 9\}$. Members of the collection Ω are shown in Table 1. Obviously, Assumption 2.1 is satisfied. Figure 1 shows the network G constructed from the instance. Let us describe the process of Algorithm 1. By pruning the out-tree of G, the vertices U_9 and U_{10} are removed from G in Step 1. Step 2 initializes C to $\{(U_1, U_4), (U_3, U_7), (U_3, U_8), (U_2, U_5), (U_2, U_6)\}$. Step 4 updates C to $\{(U_1, U_4), (U_3, U_7),$ (-$U_3, U_8), (s, U_2)\}$ once whenever U_2 is checked in the for-loop. Then, the optimal solution of problem (1) is J = {2, 4, 7, 8} with weight equal to 15.

3 CONCLUSION

In this paper, we have focused on a special cases of the weighted set cover problem. In this case, the assumption that any two sets are either disjoint or one contains the other is satisfied. It is proved that this special case is transformed into a minimum cut problem defined on an auxiliary network. Then we presented an efficient algorithm to find the minimum cut in the auxiliary network.
For future work, the methods presented for this special case can be applied to design approximation algorithms for solving other instances of the weighted set cover problem.

References

[1] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B., 1993. Network Flows. Englewood Cliffs, NJ: Prentice Hall.
[2] Bruni, R., & Montanari, U. (2017). Partial orders and fixpoints. Models of Computation (pp. 103–126). Cham: Springer International Publishing.
[3] Gainer-Dewar, A., & Vera-Licona, P. (2017). The minimal hitting set generation problem: Algorithms and computation. SIAM Journal on Discrete, 31(1), 63–100.
[4] Karp, R. M. (1972). Reducibility among combinatorial problems. Miller R.E., Thatcher J.W., Bohlinger J.D. (Ed.), Complexity of Computer Computations (pp. 85–103). Boston, MA: Springer, US.