Online Set Cover with Happiness Costs

Christine Markarian
Department of Engineering and Information Technology, University of Dubai, U.A.E.

Keywords: Combinatorial Optimization, Online Algorithms, Competitive Analysis, Online Set Cover, Happiness Cost.

Abstract: The Online Set Cover problem (OSC) and its variations are one of the most well-studied optimization problems in operations research and computer science. In OSC, we are given a universe of elements and a collection of subsets of the universe. Each subset is associated with a cost. As elements arrive over time, the algorithm purchases subsets to cover these elements. In each step, an element arrives, and the algorithm needs to ensure that at the end of the step, there is at least one purchased subset that contains the element. The goal is to minimize the total cost of purchased subsets. In this paper, we study a generalization of OSC, in which a request consisting of a number of elements arrives in each step. Each request is associated with a happiness cost. A request is served by either a single subset containing all of its elements or by a number of subsets jointly containing all of its elements. In the latter case, the algorithm needs to pay the happiness cost associated with the request. The goal is to serve all requests upon their arrival while minimizing the total cost of purchased subsets and happiness costs paid. This problem is motivated by intrinsic service-providing scenarios in which clients need not only be served but are to be satisfied with the service. Keeping clients happy by serving them with one service provider rather than many, is represented by happiness costs. We refer to this problem as Online Set Cover With Happiness Costs (OSC-HC) and design the first online algorithm, which is optimal under the competitive analysis framework. The latter is a worst-case analysis framework and the standard to measure online algorithms. It compares, for all instances of the problem, the performance of the online algorithm to that of the optimal offline algorithm that is given all the input sequence at once and is optimal.

1 INTRODUCTION

The *Set Cover* problem (SC) is one of the most well-known optimization problems, extensively studied in operations research, computer science, and combinatorics (Feige, 1998; Slavık, 1997; Feige et al., 2004; Caprara et al., 2000). It has been shown to be NP-complete in 1972 as one of Karp's 21 NP-complete problems (Karp, 1972). Given a universe of elements and a collection of subsets of the universe, each associated with a cost, SC asks to purchase subsets, of minimum costs, such that each element belongs to at least one of these subsets. SC appears in many real-world optimization scenarios, including client-server applications, in which subsets represent servers and elements represent clients that need to be served at minimum possible costs. (Vemuganti, 1998) presents a survey of applications in various areas as capital budgeting, cutting stock, scheduling, and vehicle routing.

SC and its variations have been studied in many contexts including complexity theory, approximation algorithms, and online algorithms (Feige, 1998; Alon et al., 2003; Clarkson and Varadarajan, 2007; Duh and Fürer, 1997; Shuai and Hu, 2006; Gupta et al., 2017a; Markarian and Kassar, 2020; Abshoff et al., 2016). In this paper, we continue the study of SC in the context of *online algorithms*, in which the input sequence is not given all at once but arrives in portions over time and the so-called *online algorithm* reacts to each portion as soon as it arrives while minimizing the total incurred costs. The performance of online algorithms is measured using the *competitive analysis* framework (Borodin and El-Yaniv, 2005). An online algorithm is said to be *r-competitive* or has *competitive ratio r* if the cost incurred by the algorithm, for all instances of the problem, does not exceed *r* times the cost of the optimal offline algorithm, that is informed about all the input sequence in advance and is optimal.

Many works have addressed SC in the online setting (Alon et al., 2003; Abshoff et al., 2016; Gupta et al., 2017a; Alon et al., 2005). (Alon et al., 2003) introduced the *Online Set Cover* problem (OSC), defined as follows.

Markarian, C.
Online Set Cover with Happiness Costs.
DOI: 10.5220/0010738300003062
In *Proceedings of the 2nd International Conference on Innovative Intelligent Industrial Production and Logistics (IN4PL 2021)*, pages 40-45
ISBN: 978-989-758-535-7

Definition 1. *(Online Set Cover or OSC (Alon et al., 2003)) Given a universe of elements, one arriving in each step, and a collection of subsets of the universe. Each subset is associated with a cost. As elements arrive over time, the algorithm purchases subsets to cover these elements. In each step, the algorithm needs to ensure that at the end of the step, there is at least one purchased subset that contains the arriving element. The goal is to minimize the total cost of purchased subsets.*

In this paper, we study a generalization of OSC (Alon et al., 2003), which we refer to as *Online Set Cover With Happiness Costs* (OSC-HC), defined as follows.

Definition 2. *(Online Set Cover With Happiness Costs or OSC-HC) Given a universe of n elements and a collection of m subsets of the universe. Each subset is associated with a cost. In each step, a request consisting of at most k elements arrives, such that each element arrives only once. Each request is associated with a happiness cost. A request is served by either a single subset containing all of its elements or by a number of subsets jointly containing all of its elements. In the latter case, the algorithm needs to pay the happiness cost associated with the request. At the end of each step, the algorithm needs to serve the request by previously purchased subset(s) or by purchasing new subset(s). The goal is to minimize the total cost of purchased subsets and happiness costs paid.*

OSC-HC is motivated by intrinsic service-providing scenarios in which clients need not only be served but are to be satisfied with the service. With today's digital transformation, maintaining good customer relationships becomes one of the most essential elements of a business' success and growth. Keeping clients happy by serving them with one service provider rather than many, is one way to provide a comfortable service and hence achieve customer satisfaction. We have formalized this as the *Online Set Cover With Happiness Costs* problem (OSC-HC), by introducing happiness costs associated with requests and incorporating them into the optimization objective. From an algorithmic point of view, this generalization of OSC makes sense in the presence of happiness costs since otherwise elements arriving in a single step can be treated as arriving sequentially and so any algorithm for OSC would suffice.

The *Online Set Cover* problem (OSC) due to (Alon et al., 2003) is a special case of OSC-HC in which the number k of elements for all requests is set to 1 and all happiness costs are set to 0. There are two lower bounds for OSC-HC resulting from lower bounds given for OSC. (Alon et al., 2003) showed that the best competitive ratio achievable by any deterministic algorithm for OSC is $\Omega(\frac{\log n \log m}{\log\log n + \log\log m})$, where n is the number of elements and m is the number of subsets. Furthermore, (Korman, 2005) showed that no polynomial-time randomized algorithm can achieve a competitive ratio better than $\Omega(\log n \log m)$, under the assumption that $BPP \neq NP$.

Results & Techniques. We develop the first online algorithm for OSC-HC, which we show has an asymptotically optimal $O(\log d \log n)$-competitive ratio, matching the lower bound in (Korman, 2005), where:

– d is the maximum number of subsets an element belongs to (which is at most m)

– n is the number of elements

Our algorithm is randomized and based on:

– Formulating a given instance of OSC-HC as an online directed edge-weighted graph problem with connectivity requirements

– Applying the techniques of *constructing a fractional solution* and *randomized rounding* (Raghavan and Tompson, 1987) to achieve a feasible solution for the underlying graph problem

– Mapping the graph solution to output a feasible solution for OSC-HC

Outline. The rest of the paper is structured as follows. In Section 2, we give an overview of works related to OSC-HC. In Section 3, we give a graph formulation of OSC-HC. In Section 4, we present our online algorithm and prove its competitive ratio in Section 5. We dedicate Section 6 to some thoughts about future work.

2 RELATED WORK

The *Set Cover* problem (SC) has been extensively studied in the online setting. (Alon et al., 2003) introduced the *Online Set Cover* problem (OSC) and proposed an online deterministic algorithm with nearly optimal $O(\log n \log m)$-competitive ratio, where n is the number of elements and m is the number of subsets.

(Alon et al., 2005) gave an $O(\log n \log m)$-competitive randomized algorithm for a generalization of OSC in which elements are repeated and the algorithm needs to cover a repeating element each time by a different subset. Moreover, a wide range of covering problems related to OSC were studied in the context of primal-dual algorithms (Buchbinder and Naor, 2005; Azar et al., 2016).

Another related problem appears in (Bhawalkar et al., 2014), in which requests comprise a set of elements and subsets have capacities forming packing constraints. The authors provide a randomized algorithm with nearly optimal competitive ratio. Note that although in this model, a request contains a set of elements, as in OSC-HC, the problem is substantially different.

Many other online models for SC were introduced, such as (Gupta et al., 2017b; Abshoff et al., 2016; Markarian and Kassar, 2020). (Gupta et al., 2017b) studied an online dynamic model in which elements that need to be covered change over time and the goal is to construct a solution while making as few changes per timestep as possible. (Abshoff et al., 2016; Markarian and Kassar, 2020) studied an online leasing model in which subsets are not purchased but leased for different durations and prices, and elements need to be covered only at the step they arrive.

3 GRAPH FORMULATION & SOLUTION MAPPING

In this section, we formulate the *Online Set Cover With Happiness Costs* problem (OSC-HC) as an online directed edge-weighted graph problem with connectivity requirements.

Given an instance of OSC-HC. The algorithm initially knows the universe of elements, the subsets, and the subset costs. In each step, the adversary reveals to the algorithm a request comprising a set of elements and a happiness cost. Before the appearance of any request, the algorithm constructs the following nodes and edges.

Before Any Request Arrives. The algorithm constructs for each element a so-called *element node* and for each subset, two nodes, called a *subset node* and a *duplicate subset node*, respectively. For the edges, the algorithm adds a directed edge from each subset node to its corresponding duplicate subset node, of weight equal to the cost of the associated subset. The algorithm also adds a directed edge from each duplicate subset node to each element node it contains, of weight equal to 0.

Upon the arrival of a new request, the algorithm constructs the following nodes and edges.

Whenever a Request Arrives. The algorithm constructs a *request node* for the request and a so-called *intermediary node*. For the edges, the algorithm adds a directed edge from the request node to each subset node that contains all of its elements, of weight equal to 0. It also adds a directed edge from the request node to the intermediary node, of weight equal to the happiness cost associated with the request. Moreover, it adds a directed edge from the intermediary node to each subset node that contains at least one element of the request but does not contain all of the elements, of weight equal to 0.

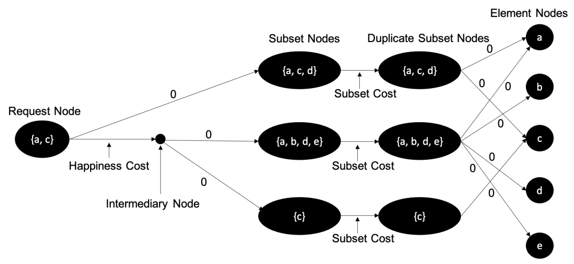

Figure 1: OSC-HC graph formulation of an instance of five elements, three subsets, and a request of two elements.

An example graph for an OSC-HC instance of five elements, three subsets, and a request of two elements is illustrated in Figure 1. The problem can now be described as follows. When a new request arrives, the algorithm needs to find a directed path from the request node to each element node comprising the request. The solution paths purchased by the algorithm would either consist of the intermediary node or not. This is translated as either a single subset covering all the elements of the request, or multiple subsets needed to cover the elements of the request. In the former case, the happiness cost, represented as the weight on the edge from the request node to the intermediary node, is paid, since this edge belongs to the solution paths. Finally, the algorithm will purchase the subsets that correspond to the edges in the solution paths purchased. That is, the weights of the edges from the subset nodes to the corresponding duplicate subset nodes that belong to the solution paths represent the subset costs incurred by the algorithm.

4 ONLINE ALGORITHM

In this section, we present an online randomized algorithm for OSC-HC based on the online graph problem described earlier.

Following the example in Figure 1, request $\{a, c\}$ arrives. The algorithm needs to find two paths: one from the request node to the element node a, and another from the request node to the element node c. Every edge of weight equal to a subset cost represents the corresponding subset. Thus, if such an edge belongs to the solution paths, then the subset is purchased by the algorithm. Similarly, every edge of weight equal to a happiness cost represents the request's happiness cost. Hence, the algorithm pays the happiness cost if the corresponding edge is in the solution paths.

Upon the arrival of a new request, the nodes and the edges associated with the request are formed, as described in the previous section. The algorithm assigns a fractional value v_e to each edge e in the graph. These values are set to 0 when the edges are formed and increase over time. The *maximum flow* between two nodes is defined as the smallest total values of edges which if removed would disconnect the two nodes. These edges form a *minimum cut*. Before the algorithm receives its first request, it generates a random number r chosen as the minimum among $2\lceil \log n \rceil$ independent random variables distributed uniformly in the interval $[0,1]$, where n is the total number of elements.

Given a request and its elements. For each element node, the algorithm calls the so-called *Find-Directed-Path* function, that returns a set of edges forming a directed path from the request node to the element node. We denote the weight of an edge e by w_e. The algorithm's steps describing its reaction to a new request are depicted in Algorithm 1 below.

Algorithm 1: Online Algorithm for OSC-HC.

```
For each element node $j$:
If there already is a directed path to the element node $j$ in the current solution, do nothing. Otherwise,
    - Run Find-Directed-Path($j$).
    - Buy the subsets and pay the happiness costs corresponding to the edges outputted by Find-Directed-Path($j$).

Find-Directed-Path($j$)
i. While the maximum flow from the request node to $j$ is less than 1:
    - Generate a minimum cut $\mathcal{K}$ from the request node to $j$ and then increase the value $v_e$ of each edge $e \in \mathcal{K}$ according to the following equation:
          $v_e \leftarrow v_e(1 + \frac{1}{w_e}) + \frac{1}{|\mathcal{K}| \cdot w_e}$
ii. Output edge $e$ if its value $v_e \geq r$.
iii. If there is no directed path to the element node $j$ in the current solution, output the edges of a smallest-weight directed path from the request node to the element node $j$.
```

Feasibility. Upon the arrival of a new request, the algorithm breaks down the request into parts, one for each element in the request, after forming the graph nodes and edges associated with the request. According to the graph formulation, the only way a request node is connected to an element node is through a directed path containing a subset node and its duplicate subset node which represent a subset containing the element in the original instance. The weight of the edge in between these nodes is what the algorithm pays in terms of subset costs. If the solution path does not contain the intermediary node, then the edge associated with a happiness cost is not in this path as per the graph construction. In this case, the algorithm does not pay a happiness cost. Note that, at some point in a given time step, if the algorithm decides to purchase a subset that contains all the elements of the request, then all the remaining elements of the request are covered and the step ends. On the other hand, even if the algorithm pays the happiness cost at some point in a given time step, it may eventually decide to purchase a subset that contains all the elements of the request. In such a case, the algorithm, in practice, does not have to pay for the happiness cost, since its decisions are revocable within a time step. However, as we will see in the next section, this does not affect our competitive analysis of the algorithm.

5 COMPETITIVE ANALYSIS

In this section, we show that the online algorithm presented has an $O(\log d \log n)$-competitive ratio, where d is the maximum number of subsets an element belongs to and n is the number of elements.

Let S be the collection of edges outputted by the algorithm excluding the edges outputted in Step iii. We denode by C_S the total weight of these edges. Let Opt be the cost of an optimal offline solution.

The function *Find-Directed-Path* outputs an edge if its fractional value exceeds the random number r (recall that r is generated before the execution of the algorithm).

Fix edge e and $i : 1 \leq i \leq 2\lceil \log n \rceil$. We denote by $X_{e,i}$ the indicator variable of the event that e is outputted by *Find-Directed-Path*. C_S can then be expressed using the following sum.

$$C_S = \sum_{e \in S} \sum_{i=1}^{2\lceil \log n \rceil} w_e \cdot Exp\left[X_{e,i}\right] \tag{1}$$

$$= 2\lceil \log n \rceil \sum_{e \in S} w_e v_e \tag{2}$$

We compare now this sum to the optimal offline solution, as follows. Every time a minimum cut is constructed, there is at least one edge that belongs to the optimal offline solution, due to the definition of minimum cut and the fact that the optimal solution must also find a directed path to each element node. It remains to count the number of times the function *Find-Directed-Path* constructs a minimum cut.

Lemma 1. *The total number of times Find-Directed-Path constructs a minimum cut is at most $O(Opt \cdot \log|K|)$, where $|K|$ is the size of the largest minimum cut.*

Proof. The proof is based on observing the edges in the optimal solution. Each edge in the optimal solution can appear in a bounded number of minimum cuts, after which its fractional value becomes 1 and it can't appear in any future minimum cut, as per the while-loop condition of the *Find-Directed-Path* function. This bound can be achieved by using the equation in the algorithm and counting the number of times needed for the fractional value of any edge in the optimal solution to become 1. The bound depends on the weight of the corresponding edge and the maximum size of the minimum cut: $O(w_e \log|K|)$. This bound holds true for every edge in the optimal solution. Moreover, each minimum cut constructed must contain at least one edge from the optimal solution. Hence, the number of times the function *Find-Directed-Path* constructs a minimum cut is $O(Opt \cdot \log|K|)$. □

We also have that the size of any minimum cut constructed does not exceed d, the maximum number of subsets an element belongs to, since there are at most d directed paths from the request node to the element node. Hence, $|K| \leq d$. Next, we show that the total increase in the fractional values of the edges associated with each minimum cut can be upper bounded by 2.

Lemma 2. *The total increase in the fractional values of the edges associated with each minimum cut is at most 2.*

Proof. Fix any minimum cut K. Edge e in K incurs an increase of $w_e \cdot \left(\frac{v_e}{w_e} + \frac{1}{|K| \cdot w_e}\right)$. The maximum flow is less than 1 before the increase is made, that is, $\sum_{e \in K} v_e < 1$. The same can be said about all the edges in the cut. Hence, the following holds true:

$$\sum_{e \in K} w_e \cdot \left(\frac{v_e}{w_e} + \frac{1}{|K| \cdot w_e}\right) < 2$$

□

As a result of Lemma 1 and Lemma 2, we conclude that $\sum_{e \in S} w_e v_e \leq O(Opt \cdot \log d)$. Thus,

$$C_S \leq O(Opt \cdot \log n \cdot \log d) \tag{3}$$

It remains to upper bound the cost of the algorithm in Step iii, which we prove affects the competitive ratio of the algorithm by a negligible factor.

We define the *flow* of a path to be the minimum value among the edge values of the path. To calculate the cost incurred in Step iii, we need to observe the probability that there is no directed path outputted for the element node prior to this step. This probability is at most the probability that r exceeds the flow of each directed path to the element node. We fix a minimum cut $\mathcal{K}$ constructed at the end of Step i. Before performing Step ii, *Find-Directed-Path* guarantees that the sum of flows of all paths to the element node is at least 1. Hence, the probability that there is no directed path to the element is:

$$\prod_{e \in \mathcal{K}} (1 - v_e) \leq e^{-\sum_{e \in \mathcal{K}} v_e} \leq \frac{1}{e}$$

Computing for all i: $1 \leq i \leq 2\lceil \log n \rceil$, the probability that there is no directed path to the element node is at most $\frac{1}{n^2}$. When there is no directed path to the element node, the algorithm buys the smallest-weight path to the element node, which is a lower bound on Opt. Thus, the cost of the algorithm incurred in Step iii is at most $n \cdot \frac{Opt}{n^2}$, since there are n elements and each element may arrive only once.

Hence, we conclude the following theorem.

Theorem 1. *There is an $O(\log d \log n)$-competitive randomized algorithm for the Online Set Cover with Happiness Costs problem, where d is the maximum number of subsets an element belongs to and n is the number of elements.*

6 CONCLUDING THOUGHTS

We have proposed in this paper a new framework for optimization problems, motivated by service-providing scenarios in which the goal is to minimize serving costs while taking into account the happiness of clients. We have considered in our model one happiness prespective, in which clients prefer to be served through one service provider. Clearly, there is much one can explore, since the satisfaction of clients can be viewed from various perspectives.

Another research direction is to study other optimization problems with different objectives, using this framework, such as the *Online Facility Location with Service Installation Costs* problem, in which the distances between clients and the facilities they are served by are also minimized (Markarian, 2021; Markarian and Khallouf, 2021).

Each element in our model arrives only once. This is in fact needed to achieve the competitive ratio of the algorithm. It would be interesting to extend our model to include repetition of elements as in (Alon et al., 2005).

In our model, the algorithm is not given any information about future requests. In fact, it might be possible to make some assumptions about the future and to use this information to improve decisions, by considering, for instance, various probability distributions for the request arrival.

Implementing the proposed algorithm on a simulated or real environment is an interesting next step. This would allow us to understand the difficulty of the problem as well as the performance of the algorithm and its effectiveness in practical applications.

REFERENCES

Abshoff, S., Kling, P., Markarian, C., auf der Heide, F. M., and Pietrzyk, P. (2016). Towards the price of leasing online. *Journal of Combinatorial Optimization*, 32(4):1197–1216.

Alon, N., Awerbuch, B., and Azar, Y. (2003). The online set cover problem. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 100–105, New York, NY, USA. Association for Computing Machinery.

Alon, N., Azar, Y., and Gutner, S. (2005). Admission control to minimize rejections and online set cover with repetitions. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 238–244.

Azar, Y., Buchbinder, N., Chan, T.-H. H., Chen, S., Cohen, I. R., Gupta, A., Huang, Z., Kang, N., Nagarajan, V., Naor, J., and Panigrahi, D. (2016). Online algorithms for covering and packing problems with convex objectives. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–157.

Bhawalkar, K., Gollapudi, S., and Panigrahi, D. (2014). Online Set Cover with Set Requests. In Jansen, K., Rolim, J. D. P., Devanur, N. R., and Moore, C., editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 64–79, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Borodin, A. and El-Yaniv, R. (2005). *Online computation and competitive analysis*. cambridge university press.

Buchbinder, N. and Naor, J. (2005). Online primal-dual algorithms for covering and packing problems. In Brodal, G. S. and Leonardi, S., editors, *Algorithms – ESA 2005*, pages 689–701, Berlin, Heidelberg. Springer Berlin Heidelberg.

Caprara, A., Toth, P., and Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–.

Clarkson, K. L. and Varadarajan, K. (2007). Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58.

Duh, R.-c. and Fürer, M. (1997). Approximation of k-set cover by semi-local optimization. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 256–264.

Feige, U. (1998). A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652.

Feige, U., Lovász, L., and Tetali, P. (2004). Approximating min sum set cover. *Algorithmica*, 40(4):219–234.

Gupta, A., Krishnaswamy, R., Kumar, A., and Panigrahi, D. (2017a). Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 537–550.

Gupta, A., Krishnaswamy, R., Kumar, A., and Panigrahi, D. (2017b). Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 537–550, New York, NY, USA. Association for Computing Machinery.

Karp, R. (1972). Reducibility among combinatorial problems. volume 40, pages 85–103.

Korman, S. (2005). On the use of randomization in the online set cover problem. Master's thesis, Weizmann Institute of Science, Israel.

Markarian, C. (2021). Online non-metric facility location with service installation costs. In Filipe, J., Smialek, M., Brodsky, A., and Hammoudi, S., editors, *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming, April 26-28, 2021, Volume 1*, pages 737–743. SCITEPRESS.

Markarian, C. and Kassar, A.-N. (2020). Online deterministic algorithms for connected dominating set & set cover leasing problems. In *ICORES*, pages 121–128.

Markarian, C. and Khallouf, P. (2021). Online facility service leasing inspired by the COVID-19 pandemic. In Gusikhin, O., Nijmeijer, H., and Madani, K., editors, *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2021, Online Streaming, July 6-8, 2021*, pages 195–202. SCITEPRESS.

Raghavan, P. and Tompson, C. D. (1987). Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374.

Shuai, T.-P. and Hu, X.-D. (2006). Connected set cover problem and its applications. In *International Conference on Algorithmic Applications in Management*, pages 243–254. Springer.

Slavık, P. (1997). A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237–254.

Vemuganti, R. R. (1998). *Applications of Set Covering, Set Packing and Set Partitioning Models: A Survey*, pages 573–746. Springer US, Boston, MA.