# Online Set Cover with Rated Subsets

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

Keywords: Online Set Cover, Rating, Optimization, Online Algorithms, Competitive Analysis, Randomized Rounding.

Abstract: In this paper, we introduce the *Online Set Cover With Rated Subsets* problem (OSC-RS), a generalization of the well-known *Online Set Cover* problem, in which we are given a universe of elements and a collection of subsets of the universe, each associated with a subset cost and a rating cost. In each step, the algorithm is given a request containing elements from the universe. The algorithm serves a request by assigning it to a number of purchased subsets that jointly cover the requested elements. The algorithm pays the subset costs associated with the subsets purchased and for each request, it pays the sum of the rating costs associated with the subsets assigned to the request. The aim is to serve all requests as soon as revealed, while minimizing the total subset and rating costs paid. OSC-RS is motivated by intrinsic client-service-providing scenarios in which service providers are rated and their ratings are included in the decision-making process, so as higher-rated service providers are associated with lower rating costs. That is, the decisions about serving clients take into account the quality of the services provided. We propose the first online algorithm for OSC-RS and evaluate it using the standard notion of *competitive analysis*. The latter compares the performance of the online algorithm to that of an optimal offline algorithm that is assumed to know all the input sequence in advance.

## 1 INTRODUCTION

In our digital world today, nearly every service or product offered is accompanied with a rating (Landy and Farr, 1980). We rely on customer reviews to make most of our day-to-day decisions, including which hotel to stay in, which restaurant to go to, and which course to register for. The customer review system is applied in almost every industry, urging companies to keep good reputation in order to succeed or even survive (Trenz and Berger, 2013; Zhou et al., 2014). Despite the vital role played by customer review systems in ensuring quality control for companies, most real-world optimization problems studied thus far in algorithmic theory do not include the rating factor in their optimization model.

In this paper, aiming to better reflect today's real-world companies, that highly rely on customer review systems, we introduce the concept of rating in the optimization objective. In particular, we address a variant of the well-known *Set Cover* (SC), a fundamental problem in operations research, computer science, and combinatorics (Feige, 1998; Slavık, 1997). SC has numerous applications in scheduling, budgeting, networks among others (Vemuganti, 1998). Given a universe of elements and a collection of subsets of the universe, each associated with a positive cost, SC asks to purchase subsets of minimum total costs that can cover the elements of the universe.

In the online setting (Borodin and El-Yaniv, 2005), elements of the universe are revealed to the online algorithm over time and the problem is known as the *Online Set Cover* problem (OSC) (Alon et al., 2009). OSC appears, among many other applications, in client-service-providing scenarios in which elements represent clients and subsets represent service providers. Covering elements is translated to serving clients and subset costs are equivalent to service providing costs, where each service provider can serve a subset of the clients.

## 2 ONLINE SET COVER WITH RATED SUBSETS (OSC-RS)

In this paper, we study the *Online Set Cover With Rated Subsets* problem (OSC-RS), a generalization of OSC, in which we are given a universe of $n$ elements and a collection of $m$ subsets of the universe, each associated with a positive subset cost and a positive rating cost. In each step, the algorithm is given a request containing at most $k$ elements from the universe. Each element may belong to at most $d$ subsets. The algorithm serves a request by assigning it to a number of purchased subsets that jointly cover the re-

Markarian, C.
Online Set Cover With Rated Subsets.
DOI: 10.5220/0010976900003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 1*, pages 455-461
ISBN: 978-989-758-569-2; ISSN: 2184-4992

quested elements. The algorithm pays the subset costs associated with the subsets purchased and for each request, it pays the sum of the rating costs associated with the subsets assigned to the request. The aim is to serve all requests as soon as revealed, while minimizing the total subset and rating costs paid. Note that in OSC (Alon et al., 2009), a single element arrives in each time step. Hence, OSC-RS generalizes OSC by setting the number of elements per request to 1 and all rating costs to 0.

OSC-RS is motivated by intrinsic client-service-providing scenarios in which service providers are rated and their ratings are included in the decision-making process, so as higher-rated service providers are associated with lower rating costs. That is, the higher the rating of a service provider, the lower is its rating cost. Consider a company that assigns service providers to serve its clients. Unlike in the *Online Set Cover* problem, in OSC-RS, the decisions about serving clients take into account the quality of the services provided. Hence, the company does not only aim to minimize its classical costs, but considers low-rated services to be additional costs that represent its reputation-maintainance costs. In OSC-RS, this is reflected in the rating costs associated with the subsets in addition to the subset costs.

## 2.1 Our Contribution

We propose the first online algorithm for OSC-RS and evaluate it using the standard notion of *competitive analysis*, which is defined as follows. An online algorithm has competitive ratio $c$ (or is $c$-competitive) if for all input sequences, the cost incurred by the online algorithm is at most $c$ times the cost of an optimal offline algorithm that knows the entire input sequence in advance and is optimal. The input to an online algorithm is revealed in portions over time, and the algorithm is expected to react to each portion while targeting the given optimization goal against the whole input sequence. Unlike offline algorithms that react once to a given input sequence, the challenge of an online algorithm is to react to each of the input portions as soon as revealed, without knowing about the remaining input sequence in advance.

## 2.2 Relationships with Other Problems

The special case of OSC-RS in which each request is composed of one element, can be solved by transforming a given instance of the problem into an instance of the *Non-metric Online Facility Location* problem (non-metric OFL) (Alon et al., 2006), defined as follows. Given a collection of potential facility locations. Each facility can be opened by paying its so-called opening cost. Clients arrive at different locations over time. Non-metric OFL asks to connect each arriving client to an open facility. Connecting a client to an open facility incurs the so-called connecting cost between the client and the facility, which is the distance between the client location and the facility location. The transfomation works as follows. Each subset is formed as a facility location and facility opening costs correspond to subset costs. Each element is formed as a client and the connecting cost between a client and a facility is set to the rating cost of the subset corresponding to the facility. If an element does not belong to a subset, the connecting cost between the corresponding client and the corresponding facility is set to infinity. Every time a client is connected to a facility, the corresponding subset's rating cost is paid.

Our algorithm for OSC-RS which we describe in the coming sections will handle requests composed of *any* bounded number of elements.

As mentioned earlier, OCS-RC generalizes the *Online Set Cover* problem (OSC) due to (Alon et al., 2009), in which the number $k$ of elements in each request is 1 and all rating costs are 0.

– There is $\Omega(\frac{\log n \log m}{\log\log n + \log\log m})$ lower bound on the competitive ratio of any deterministic algorithm for OSC, where $n$ is the number of elements and $m$ is the number of subsets, due to (Alon et al., 2009).

– There is $\Omega(\log n \log m)$ lower bound on the competitive ratio of any randomized polynomial-time algorithm for OSC, under the assumption that $BPP \neq NP$, due to (Korman, 2005).

## 2.3 Competitive Ratio

Our algorithm for OSC-RS has an $O(\log d \log n)$ competitive ratio, where:

– $d$ is the maximum number of subsets any element can belong to.

– $n$ is the total number of elements.

## 2.4 Algorithmic Techniques

Our online algorithm is based on *randomized rounding*, a technique used to solve many online optimization problems (Markarian, 2021; Markarian et al., 2021; Hamann et al., 2018; Markarian, 2018; Alon et al., 2006). Moreover, the algorithm is based on a transformation of a given OSC-RS instance into a graph-theoretic problem instance, as input portions appear over time.

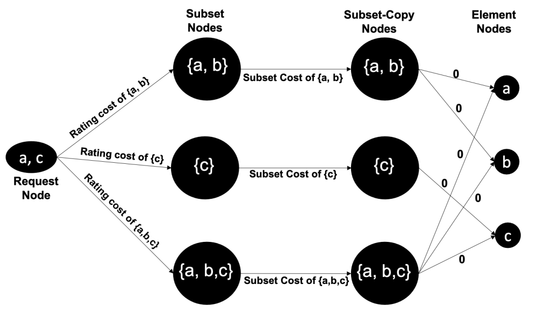

Figure 1: OSC-RS graph instance of three elements, three subsets, and a request of two elements.

## 3 TRANSFORMATION OF OSC-RS INSTANCE

Given an instance $I$ of the *Online Set Cover With Rated Subsets* problem (OSC-RS). We transform $I$ into a directed edge-weighted graph problem instance, as follows. We refer the reader to Figure 1 for an illustration.

**The Nodes of the Graph Are Formed as Follows:**

- We associate each element in the universe with an *element node*.
- We associate each subset with two nodes: a *subset node* and a *subset-copy node*.
- We associate each request with a *request node*.

**The Edges of the Graph Are Formed as Follows:**

- We add a directed edge from each subset-copy node to each element node if the corresponding subset contains the corresponding element. These edges have weight 0.
- We add a directed edge from each subset node to its corresponding subset-copy node, of weight equal to the corresponding subset cost.
- For each request, we add a directed edge from the request node to each subset node which corresponds to a subset containing at least one element comprising the request. The weight of each outgoing edge to a subset node will be equal to the rating cost of the corresponding subset.

Recall that the goal of the algorithm is to assign each request to a number of subsets that jointly cover the elements of the request. On a given time step, the algorithm may purchase new subsets by paying their subset costs. This goal is tansformed in our graph problem as follows.

Initially, the subset nodes, the subset-copy nodes, and the element nodes are formed, including the weighted edges between them, as described above.

In each step, as soon as a new request arrives:

- The algorithm forms a request node associated with the request and adds all the outgoing edges from it, as described above.
- The algorithm finds a directed path from the request node to each of the element nodes corresponding to the elements of the request. To do this, the algorithm purchases edges of the graph that form the desired paths.
- The algorithm pays the subset costs and the rating costs associated with the purchased edges, as follows. Whenever the algorithm purchases an edge that connects a subset node to its subset-copy node, the corresponding subset cost is paid. Whenever the algorithm purchases an edge connecting a request node to a subset node, the corresponding rating cost of the subset is paid.

It is easy to see that a feasible solution for the graph instance formed from a given instance $I$ of OSC-RS according to the transformation above, implies a feasible solution for $I$, of the same cost.

## 4 ONLINE ALGORITHM

In this section, we present an online algorithm that finds a feasible solution for the graph problem described in the previous section. This implies a feasible solution for any given instance of OSC-RS.

The algorithm is randomized and has two phases. In the first phase, the algorithm uses a randomized approach to purchase edges from the graph, that may form an infeasible solution. In the second phase, the algorithm may buy further edges to guarantee a feasible solution.

This idea has been implemented in other algorithms for similar online problems (Markarian, 2021; Markarian et al., 2021; Hamann et al., 2018; Markarian, 2018; Alon et al., 2006).

A fractional value $v_e$ is assigned to each edge $e$ in the graph, all set to 0 initially and non-decreasing throughout the execution of the algorithm. The concepts of maximum flow and minimum cut will be used. The *maximum flow* from node $i$ to node $j$ is the smallest total fractional values of edges which would break the directed path from $i$ to $j$ if removed. These edges are called *minimum cut* edges.

The algorithm chooses a random value $q$, independently among $2\lceil \log n \rceil$ random variables that are uniformly distributed in the interval $[0,1]$, where $n$ is the total number of elements. This value $q$ will be used by the algorithm in its first phase when choosing the edges.

Recall that the edges of the graph are directed and weighted, as per the transformation described in the previous section. We refer to the weight of an edge $e$ by $w_e$.

The online algorithm reacts to each new element $s$ in a given request as follows:

If the algorithm's current outputted solution does not contain a directed path from the request node to the element node associated with $s$, the algorithm performs the following:

**Phase 1.** The algorithm calculates the maximum flow from the request node to the element node associated with $s$. As long as this value does not exceed 1, the algorithm constructs a minimum cut $Q$ and increases the fractional value of each edge in $Q$, based on its weight and the cardinality of $Q$, using the equation below:

$$v_e \leftarrow v_e(1+\frac{1}{w_e})+\frac{1}{|Q|\cdot w_e}$$

When the maximum flow is at least 1, the algorithm adds each edge whose fractional value exceeds $q$ (the random value described earlier) to the solution.

**Phase 2.** At this point, the algorithm might have constructed an infeasible solution - that depends on the randomization process. To ensure a feasible solution, the algorithm observes the edges in the solution so far outputted. If the solution does not contain a directed path from the request node to the element node associated with $s$, the algorithm calculates a shortest weighted directed path from the request node to the element node and adds the edges of this path to the solution.

The two phases of the algorithm are depicted in Algorithm 1.

## 5 COMPETITIVE ANALYSIS

In this section, we prove an upper bound on the competitive ratio of the algorithm presented in the previous section.

As mentioned in Section 3, the expected cost of the solution for a given instance $I$ of OSC-RS is less or equal to the expected solution cost of the corresponding graph instance. Hence, to analyze the competitive ratio of the algorithm, we observe the total expected cost of edges purchased by the algorithm in comparison to that of an optimal offline solution.

Algorithm 1: Online Algorithm for OSC-RC.

```
Phase 1.
– While the maximum flow from the request node to
  the element node associated with $s$ is less than 1:

      Construct a minimum cut $Q$ from the request
  node to the element node associated with $s$ and in-
  crease the fractional value $v_e$ of each edge $e \in Q$,
  using the equation below:

      $v_e \leftarrow v_e(1+\frac{1}{w_e})+\frac{1}{|Q|\cdot w_e}$

– Add each edge $e$ to the solution if its value $v_e$ ex-
  ceeds $q$.
Phase 2.
  If there is no directed path from the request node
  to the element node associated with $s$ in the current
  solution, add to the solution the edges of a smallest
  weighted path from the request node to the element
  node associated with $s$.
```

Notice that the algorithm may purchase edges in the first and second phases. Let $S_1$ and $S_2$ denote the the edges purchased by the algorithm in Phase 1 and Phase 2, respectively. We will analyze the expected cost of each collection separately. We denote by Opt the cost of an optimal offline solution.

**Phase 1 - Analysis.** In the first phase, the algorithm purchases an edge whenever the fractional value associated with it becomes at least $q$ (recall the number generated by a random process before the arrival of any request). Let us observe an edge $e$ and $i : 1 \leq i \leq 2\lceil \log n \rceil$. Let $X_{e,i}$ be the variable indicating whether $e$ has been purchased by the algorithm or not. Recall that $w_e$ and $v_e$ denote the weight and fractional value of an edge $e$, respectively. We can now express the expected cost $C_{S_1}$ of the collection $S_1$ as follows:

$$C_{S_1} = \sum_{e\in S_1}\sum_{i=1}^{2\lceil \log n\rceil} w_e \cdot Exp\left[X_{e,i}\right] \tag{1}$$

$$= 2\lceil \log n\rceil \sum_{e\in S_1} w_e v_e \tag{2}$$

To make a comparison to the cost of the optimal offline solution, we need to observe the minimum cuts constructed, since the algorithm purchases edges of minimum cuts constructed in the first phase. The first observation which allows the comparison is that each minimum cut generated by the algorithm contains at least one edge that is in the optimal offline solution.

**Observation 1.** *Each minimum cut constructed by the algorithm contains at least one optimal edge.*

This observation is true because of the definition of minimum cut and since any optimal algorithm is supposed to output a directed feasible path for the given element. The second observation which allows the comparison is the number of times the algorithm constructs a minimum cut.

**Observation 2.** *The algorithm constructs $O(Opt \cdot \log|Q|)$ minimum cuts, where $|Q|$ is the size of the biggest minimum cut constructed.*

*Proof.* Observe the optimal edges. We show that each such edge would show up in a bounded number of minimum cuts, since its fractional value will eventually become 1 and so it won't show up in any later minimum cut, according to the algorithm's design.

Let us fix any optimal edge and look at the equation of the algorithm to check how many times we need to perform a fractional increase until the fractional value of this edge becomes 1. Notice that $O(w_e \log|Q|)$ fractional increases are needed to make the value of the edge become at least 1. On the other hand, as per the first observation, each minimum cut contains at least one optimal edge. Therefore, the algorithm constructs $O(Opt \cdot \log|Q|)$ minimum cuts, where $|Q|$ is the size of the biggest minimum cut constructed. □

To achieve the desired upper bound, we will need to give an upper bound on $|Q|$. Since each element may belong to at most $d$ subsets, then there are at most $d$ directed paths from the request node to each element node. Therefore, $|Q| \le d$ and so the number of minimum cuts constructed is upper bounded by $O(Opt \cdot \log|d|)$.

The next thing to look at is the total fractional increase associated with each minimum cut. Since we have an upper bound on the number of minimum cuts constructed, we can conclude an upper bound on the expected cost $C_{S_1}$ of the algorithm in Phase 1.

**Lemma 1.** *Each minimum cut constructed is associated with a maximum of 2 fractional increase.*

*Proof.* We observe any minimum cut $Q$. Each edge $e$ in $Q$ causes a fractional increase of $w_e \cdot \left(\frac{v_e}{w_e} + \frac{1}{|Q| \cdot w_e}\right)$, based on the algorithm's equation. Notice that before an increase is made, the maximum flow is less than 1. Hence, $\sum_{e \in Q} v_e < 1$. Hence, we conclude the following.

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

□

Therefore, we have that $\sum_{e \in S} w_e v_e \le O(Opt \cdot \log d)$. Thus,

$$C_{S_1} \le O(Opt \cdot \log n \cdot \log d) \tag{3}$$

**Phase 2 - Analysis.** We now bound the expected cost $C_{S_2}$ of the collection $S_2$ of edges purchased by the algorithm in Phase 2.

The *flow* of a path will be the minimum value among the edge values of a given path. We fix an element node $u$ and an $i$: $1 \le i \le 2\lceil \log n \rceil$. We calculate the probability that there is no feasible path purchased by the algorithm in Phase 1 for $u$, for a single $i$. In terms of flow, the latter is the probability that $q$ is more than the flow of each directed path to $u$. Let $Q$ be a minimum cut constructed at the end of Phase 1. Before executing Phase 2, we have that the sum of flow associated with all paths to $u$ is at least 1 according to the algorithm. Therefore, the probability that there is no feasible path purchased by the algorithm in Phase 1 for $u$, considering a single $i$, is:

$$\prod_{e \in Q} (1 - v_e) \le e^{-\sum_{e \in Q} v_e} \le \frac{1}{e}$$

Now we calculate the probability considering all $i$: $1 \le i \le 2\lceil \log n \rceil$ and conclude the following. The probability that there is no feasible path purchased by the algorithm in Phase 1 for $u$ is at most $\frac{1}{n^2}$.

Hence, the algorithm purchases a smallest weighted path from the request node to $u$ with probability at most $\frac{1}{n^2}$. Obviously, this path is upper bounded by Opt, since it is a smallest weighted path.

Since we have a total of $n$ elements, each arriving exactly once, we can conclude the following.

$$C_{S_2} \le n \cdot \frac{Opt}{n^2}$$

Therefore, we following theorem holds.

**Theorem 1.** *There is an $O(\log d \log n)$-competitive randomized algorithm for the Online Set Cover With Rated Subsets problem, where d is the maximum number of subsets any element can belong to, and n is the number of elements.*

## 6 CONCLUDING THOUGHTS

We have initiated in this paper the study of the impact of rated resources on decision-making. We have generalized the well known *Online Set Cover* problem (OSC) by associating each subset with a price that depends on the rating of the subset or resource it represents. In other words, upon purchasing a resource, the decision-maker pays in addition to the actual cost of the resource, the price for good quality service.

Our simple yet effective model is one way to represent resource ratings and relate them to the optimization objective. There are different ways this can be done, for example, rather than paying the sum of the rating costs for each subset serving a request, one could pay the average rating cost of these subsets instead.

Also, each request could be associated with an amount of money one is willing to pay to get a higher rated service for the request. In many real-world business scenarios, customers are often given options to choose from when it comes to quality of services. Hence, the decision maker would take into account the quality of service requested when assigning resources to clients. For instance, a rating cost would not have to be paid if the request is not made for a high-rated service.

Furthermore, the ratings in our model are assumed to be fixed throughout time, which is not the case in actual rating systems. One may want to include dynamic pricing for these ratings. From an algorithmic perspective, it could be that extensions of the existing algorithm would solve the variants that arise from dynamic pricing - a similar study as the one in (Feldkord et al., 2017) for leasing problems.

We have initiated this study by targeting OSC. There are a lot of other well-studied online resource allocation problems to explore. Many of these would serve as real-world decision-making problems in the context of rating, such as variants of the *Online Facility Location* problem (Alon et al., 2006; Meyerson, 2001; Markarian et al., 2021; Markarian and Khallouf, 2021; Markarian and auf der Heide, 2019) and variants of the *Online Connected Dominating Set* problem (Hamann et al., 2018; Markarian and Kassar, 2020).

## REFERENCES

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. (2006). A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660.

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. (2009). The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370.

Borodin, A. and El-Yaniv, R. (2005). *Online computation and competitive analysis*. Cambridge University Press.

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

Feldkord, B., Markarian, C., and auf der Heide, F. M. (2017). Price fluctuation in online leasing. In Gao, X., Du, H., and Han, M., editors, *Combinatorial Optimization and Applications - 11th International Conference, COCOA 2017, Shanghai, China, December 16-18, 2017, Proceedings, Part II*, volume 10628 of *Lecture Notes in Computer Science*, pages 17–31. Springer.

Hamann, H., Markarian, C., auf der Heide, F. M., and Wahby, M. (2018). Pick, pack, & survive: Charging robots in a modern warehouse based on online connected dominating sets. In Ito, H., Leonardi, S., Pagli, L., and Prencipe, G., editors, *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*, volume 100 of *LIPIcs*, pages 22:1–22:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

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

Landy, F. J. and Farr, J. L. (1980). Performance rating. *Psychological bulletin*, 87(1):72.

Markarian, C. (2018). An optimal algorithm for online prize-collecting node-weighted steiner forest. In Iliopoulos, C. S., Leong, H. W., and Sung, W., editors, *Combinatorial Algorithms - 29th International Workshop, IWOCA 2018, Singapore, July 16-19, 2018, Proceedings*, volume 10979 of *Lecture Notes in Computer Science*, pages 214–223. Springer.

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 auf der Heide, F. M. (2019). Online algorithms for leasing vertex cover and leasing non-metric facility location. In Parlier, G. H., Liberatore, F., and Demange, M., editors, *Proceedings of the 8th International Conference on Operations Research and Enterprise Systems, ICORES 2019, Prague, Czech Republic, February 19-21, 2019*, pages 315–321. SciTePress.

Markarian, C. and Kassar, A. (2020). Online deterministic algorithms for connected dominating set & set cover leasing problems. In Parlier, G. H., Liberatore, F., and Demange, M., editors, *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems, ICORES 2020, Valletta, Malta, February 22-24, 2020*, pages 121–128. SCITEPRESS.

Markarian, C., Kassar, A., and Yunis, M. M. (2021). An algorithmic approach to online multi-facility location problems. In Parlier, G. H., Liberatore, F., and Demange, M., editors, *Proceedings of the 10th International Conference on Operations Research and Enterprise Systems, ICORES 2021, Online Streaming, February 4-6, 2021*, pages 29–35. SCITEPRESS.

Markarian, C. and Khallouf, P. (2021). Online metric facility service leasing with duration-specific dormant fees. In Panetto, H., Macchi, M., and Madani, K., editors, *Proceedings of the 2nd International Conference* *on Innovative Intelligent Industrial Production and Logistics, IN4PL 2021, October 25-27, 2021*, pages 25–31. SCITEPRESS.

Meyerson, A. (2001). Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431.

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

Trenz, M. and Berger, B. (2013). Analyzing online customer reviews-an interdisciplinary literature review and research agenda.

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

Zhou, L., Ye, S., Pearce, P. L., and Wu, M.-Y. (2014). Refreshing hotel satisfaction studies by reconfiguring customer review data. *International journal of hospitality management*, 38:1–10.