IPSJ SIG Technical Report

Vol.2018-AL-170 No.14
2018/11/13

# Multi-Pass Streaming Algorithms for Monotone Submodular Function Maximization

Chien-Chung Huang[1,a)] Naonori Kakimura[2,b)]

**Abstract:** We consider maximizing a monotone submodular function under a cardinality constraint or a knapsack constraint in the streaming setting. In particular, the elements arrive sequentially and at any point of time, the algorithm has access to only a small fraction of the data stored in primary memory. We propose the following streaming algorithms taking $O(\varepsilon^{-1})$ passes:
( 1 ) a $(1 - e^{-1} - \varepsilon)$-approximation algorithm for the cardinality-constrained problem
( 2 ) a $(0.5 - \varepsilon)$-approximation algorithm for the knapsack-constrained problem.
Both of our algorithms run deterministically in $O^*(n)$ time, using $O^*(K)$ space, where $n$ is the size of the ground set and $K$ is the size of the knapsack. Here the term $O^*$ hides a polynomial of $\log K$ and $\varepsilon^{-1}$. Our streaming algorithms can also be used as fast approximation algorithms. In particular, for the cardinality-constrained problem, our algorithm takes $O(n\varepsilon^{-1}\log(\varepsilon^{-1}\log K))$ time, improving on the algorithm of Badanidiyuru and Vondrák that takes $O(n\varepsilon^{-1}\log(\varepsilon^{-1}K))$ time.

**Keywords:** Submodular function, Streaming algorithms, Approximation algorithms

## 1. Introduction

A set function $f : 2^E \to \mathbb{R}_+$ on a ground set $E$ is *submodular* if it satisfies the *diminishing marginal return property*, i.e., for any subsets $S \subseteq T \subsetneq E$ and $e \in E \setminus T$,

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

A function is *monotone* if $f(S) \leq f(T)$ for any $S \subseteq T$. Submodular functions play a fundamental role in combinatorial optimization, as they capture rank functions of matroids, edge cuts of graphs, and set coverage, just to name a few examples. In addition to their theoretical interests, submodular functions have attracted much attention from the machine learning community because they can model various practical problems such as online advertising [2], [20], [32], sensor location [21], text summarization [25], [26], and maximum entropy sampling [23].

Many of the aforementioned applications can be formulated as the maximization of a monotone submodular function under a knapsack constraint. In this problem, we are given a monotone submodular function $f : 2^E \to \mathbb{R}_+$, a size function $c : E \to \mathbb{N}$, and an integer $K \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of positive integers. The problem is defined as

$$\text{maximize } \ f(S) \quad \text{subject to } \ c(S) \leq K, \quad S \subseteq E, \tag{1}$$

where we denote $c(S) = \sum_{e \in S} c(e)$ for a subset $S \subseteq E$. Throughout this paper, we assume that every item $e \in E$ satisfies $c(e) \leq K$ as otherwise we can simply discard it. Note that, when $c(e) = 1$ for every item $e \in E$, the constraint coincides with a cardinality constraint:

$$\text{maximize } \ f(S) \quad \text{subject to } \ |S| \leq K, \quad S \subseteq E. \tag{2}$$

The problem of maximizing a monotone submodular function under a knapsack or a cardinality constraint is classical and well-studied [15], [34]. The problem is known to be NP-hard but can be approximated within the factor of $1 - e^{-1}$ (or $1 - e^{-1} - \varepsilon$); see e.g., [4], [11], [16], [22], [33], [35]. Notice that for both problems, it is standard to assume that a value oracle of a function $f$ is given and the complexity of the algorithms is measured based on the number of oracle calls.

In this work, we study the two problems with a focus on designing *space and time efficient* approximation algorithms. In particular, we assume the *streaming* setting: each item in the ground set $E$ arrives sequentially, and we can keep only a small number of the items in memory at any point. This setting renders most of the techniques in the literature ineffective, as they typically require random access to the data.

In this extended abstract, most of the details are omitted, which can be found in the full version [18].

## 2. Our contribution

Our contributions are summarized as follows.

**Theorem 2.1 (Cardinality Constraint)** Let $n = |E|$. For the problem (2), we design streaming $(1 - e^{-1} - \varepsilon)$-approximation algorithms requiring either (1) $O(\varepsilon^{-1}\log(\varepsilon^{-1}\log K))$ passes, $O(K)$ space, and $O\left(n\varepsilon^{-1}\log(\varepsilon^{-1}\log K)\right)$ running time, or (2) $O(\varepsilon^{-1})$ passes, $O\left(K\varepsilon^{-1}\log K\right)$ space, and $O\left(n\varepsilon^{-1}\log K + n\varepsilon^{-2}\right)$ running time.

[1] CNRS, École Normale Supérieure
[2] Department of Mathematics, Keio University
[a)] villars@gmail.com
[b)] kakimura@math.keio.ac.jp

**Theorem 2.2 (Knapsack Constraint)** Let $n = |E|$. We design streaming $(0.5 - \varepsilon)$-approximation algorithms for the problem (1) requiring $O\left(K\varepsilon^{-7}\log^2 K\right)$ space, $O(\varepsilon^{-1})$ passes, and $O\left(n\varepsilon^{-8}\log^2 K\right)$ running time.

To put our results in a better context, we list related work in Tables **Table 1** and **Table 2**. For the cardinality-constrained problem, our first algorithm achieves the same ratio $1 - e^{-1} - \varepsilon$ as Badanidiyuru and Vondrák [4], using the same space, while strictly improving on the running time and the number of passes. The second algorithm improves further the number of passes to $O(\varepsilon^{-1})$, which is independent of $K$ and $n$, but slightly loses out in the running time and the space requirement.

For the knapsack-constrained problem, our algorithm gives the best ratio so far using only small space (though at the cost of using more passes than [17], [36]). In the non-streaming setting, Sviridenko [33] gave a $(1-e^{-1})$-approximation algorithm, which takes $O(Kn^4)$ time. Very recently, Ene and Nguyen [12] gave a $(1-e^{-1}-\varepsilon)$-approximation algorithm, which takes $O((1/\varepsilon)^{O(1/\varepsilon^4)}n\log n)$ time[*1].

## 3. Our Technique

We first give an algorithm, called Simple, for the cardinality-constrained problem (2). This algorithm is later used as a subroutine for the knapsack-constrained problem (1). The basic idea of Simple is similar to those in [4], [29]: in each pass, a certain threshold is set; items whose marginal value exceeds the threshold are added into the collection; others are just ignored. In [4], [29], the threshold is decreased in a conservative way (by the factor of $1 - \varepsilon$) in each pass. In contrast, we adjust the threshold *dynamically*, based on the $f$-value of the current collection. We show that, after $O(\varepsilon^{-1})$ passes, we reach a $(1 - e^{-1} - \varepsilon)$-approximation. To set the threshold, we need a prior estimate of the optimal value, which we show can be found by a preprocessing step requiring either $O(K\varepsilon^{-1}\log K)$ space and a single pass, or $O(K)$ space and $O(\varepsilon^{-1}\log(\varepsilon^{-1}\log K))$ passes. The implementation and analysis of the algorithm are very simple.

For the knapsack-constrained problem (1), let us first point out the challenges in the streaming setting. The techniques achieving the best ratios in the literature are in [12], [33]. In [33], *partial enumeration* and *density greedy* are used. In the former, small sets (each of size at most 3) of items are guessed and for each guess, density greedy adds items based on the decreasing order of marginal ratio (i.e., the marginal value divided by the item size). To implement density greedy in the streaming setting, large number of passes would be required. In [12], partial enumeration is replaced by a more sophisticated multi-stage guessing strategies (where fractional items are added based on the technique of multilinear extension) and a "lazy" version of density greedy is used so as to keep down the time complexity. This version of density greedy nonetheless requires a priority queue to store the density of all items, thus requiring large space.

We present algorithms, in increasing order of sophistication, that give $0.39 - \varepsilon$, $0.46 - \varepsilon$, and $0.5 - \varepsilon$ approximations respectively. The first simpler algorithms are useful for illustrating the main ideas and also are used as subroutines for later, more involved algorithms. The first algorithm adapts the algorithm Simple for the cardinality-constrained case. We show that Simple still performs well if all items in the optimal solution (henceforth denoted by OPT) are small in size. Therefore, by ignoring the largest optimal item $o_1$, we can obtain a $(0.39 - \varepsilon)$-approximate solution.

The difficulty arises when $c(o_1)$ is large and the function value $f(o_1)$ is too large to be ignored. To take care of such a large-size item, we first aim at finding a good item $e$ whose size approximates that of $o_1$, using a single pass [17]. This item $e$ satisfies the following properties: (1) $f(e)$ is large, (2) the marginal value of OPT $- o_1$ with respect to $e$ is large. Then, after having this item $e$, we apply Simple to pack items in OPT $- o_1$. Since the largest item size in OPT $- o_1$ is smaller, the performance of Simple is better than just applying Simple to the original instance. The same argument can be applied for OPT $- o_1 - o_2$, where $o_2$ is the second largest item. These solutions, together with $e$, yield a $(0.46 - \varepsilon)$-approximation.

The above strategy would give a $(0.5 - \varepsilon)$-approximation if $f(o_1)$ is large enough. When $f(o_1)$ is small, we need to generalize the above ideas further. We propose a two-phase algorithm. In Phase 1, an initial *good set* $Y \subseteq E$ is chosen (instead of a single good item); in Phase 2, the algorithm packs items in some subset OPT$' \subseteq$ OPT using the remaining space. Ideally, the good set $Y$ should satisfy the following properties: (1) $f(Y)$ is large, (2) the marginal value of OPT$'$ with respect to $Y$ is large, and (3) the remaining space, $K - c(Y)$, is sufficiently large to pack items in OPT$'$. To find a such a set $Y$, we design two strategies, depending on the sizes, $c(o_1)$, $c(o_2)$ of the two largest items in OPT.

The first case is when $c(o_1) + c(o_2)$ is large. As mentioned above, we may assume that $f(o_1)$ is small. In a similar way, we can show that $f(o_2)$ is small. Then there exists a "dense" set of small items in OPT, i.e., $\frac{f(\mathrm{OPT}\setminus\{o_1,o_2\})}{c(\mathrm{OPT}\setminus\{o_1,o_2\})}$ is large. The good set $Y$ thus can be small items approximating $f(\mathrm{OPT} \setminus \{o_1, o_2\})$ while still leaving enough space for Phase 2.

The other case is when $c(o_1) + c(o_2)$ is small. In this case, we apply a modified version of Simple to obtain a good set $Y$. The modification allows us to lower-bound the marginal value of OPT$'$ with respect to $Y$. Furthermore, we can show that $Y$ is already a $(0.5 - \varepsilon)$-approximation when $c(Y)$ is large. Thus we may assume that $c(Y)$ is small, implying that we have still enough space to pack items in OPT$'$ in Phase 2.

For more details, see the full version of the manuscript [18].

## 4. Related Work

Maximizing a monotone submodular function subject to various constraints is a subject that has been extensively studied in the literature. We do not attempt to give a complete survey here and just highlight the most relevant results. Besides a knapsack constraint or a cardinality constraint mentioned above, the problem has also been studied under (multiple) matroid constraint(s), $p$-system constraint, multiple knapsack constraints. See [6], [8], [9], [11], [14], [22], [24] and the references therein.

[*1] In [4], a $(1 - e^{-1} - \varepsilon)$-approximation algorithm of running time $O(n^2(\varepsilon^{-1}\log\frac{n}{\varepsilon})^{\varepsilon^{-8}})$ was claimed. However, this algorithm seems to require some assumption on the curvature of the submodular function. See [12], [35] for details on this issue.

**Table 1** The cardinality-constrained problem. The algorithms [4], [16], [30] are originally not for the streaming setting.

| | approx. ratio | # passes | space | running time |
|---|---|---|---|---|
| Badanidiyuru *et al.* [3] | $0.5-\varepsilon$ | 1 | $O\left(K\varepsilon^{-1}\log K\right)$ | $O\left(n\varepsilon^{-1}\log K\right)$ |
| **Ours** | $1-e^{-1}-\varepsilon$ | $O\left(\varepsilon^{-1}\right)$ | $O\left(K\varepsilon^{-1}\log K\right)$ | $O\left(n\varepsilon^{-1}\log K+n\varepsilon^{-2}\right)$ |
| **Ours** | $1-e^{-1}-\varepsilon$ | $O\left(\varepsilon^{-1}\log\left(\varepsilon^{-1}\log K\right)\right)$ | $O(K)$ | $O\left(n\varepsilon^{-1}\log\left(\varepsilon^{-1}\log K\right)\right)$ |
| Badanidiyuru–Vondrák [4] | $1-e^{-1}-\varepsilon$ | $O\left(\varepsilon^{-1}\log(\varepsilon^{-1}K)\right)$ | $O(K)$ | $O\left(n\varepsilon^{-1}\log(\varepsilon^{-1}K)\right)$ |
| Mirzasoleiman *et al.* [30] | $1-e^{-1}-\varepsilon$ (in expectation) | $K$ | $O(K)$ | $O\left(n\log\varepsilon^{-1}\right)$ |
| Greedy [16] | $1-e^{-1}$ | $K$ | $O(K)$ | $O(nK)$ |

**Table 2** The knapsack-constrained problem. The algorithms [12], [33] are not for the streaming setting. See also [11], [22].

| | approx. ratio | # passes | space | running time |
|---|---|---|---|---|
| Yu *et al.* [36] | $1/3-\varepsilon$ | 1 | $O\left(K\varepsilon^{-1}\log K\right)$ | $O\left(n\varepsilon^{-1}\log K\right)$ |
| Huang *et al.* [17] | $0.363-\varepsilon$ | 1 | $O\left(K\varepsilon^{-4}\log^4 K\right)$ | $O\left(n\varepsilon^{-4}\log^4 K\right)$ |
| Huang *et al.* [17] | $0.4-\varepsilon$ | 3 | $O\left(K\varepsilon^{-4}\log^4 K\right)$ | $O\left(n\varepsilon^{-4}\log^4 K\right)$ |
| **Ours** | $0.39-\varepsilon$ | $O\left(\varepsilon^{-1}\right)$ | $O\left(K\varepsilon^{-2}\log K\right)$ | $O\left(n\varepsilon^{-1}\log K+n\varepsilon^{-3}\right)$ |
| **Ours** | $0.46-\varepsilon$ | $O\left(\varepsilon^{-1}\right)$ | $O\left(K\varepsilon^{-4}\log K\right)$ | $O\left(n\varepsilon^{-5}\log K\right)$ |
| **Ours** | $0.5-\varepsilon$ | $O\left(\varepsilon^{-1}\right)$ | $O\left(K\varepsilon^{-7}\log^2 K\right)$ | $O\left(n\varepsilon^{-8}\log^2 K\right)$ |
| Ene and Nguyen [12] | $1-e^{-1}-\varepsilon$ | — | — | $O\left((1/\varepsilon)^{O(1/\varepsilon^4)}n\log n\right)$ |
| Sviridenko [33] | $1-e^{-1}$ | — | — | $O\left(Kn^4\right)$ |

In the streaming setting, single-pass algorithms have been proposed for the problem with matroid constraints [7], [13] and knapsack constraint [17], [36], and without monotonicity [10], [31]. On the other hand, multi-pass streaming algorithms have not been well studied, except for [4], [7], [17]. Chakrabarti and Kale [7] gave an $O(\varepsilon^{-3})$-pass streaming algorithms for a generalization of the maximum matching problem and the submodular maximization problem with cardinality constraint. We remark that the maximum matching problem is one of the central topic in the streaming setting, and multi-pass streaming algorithms have been developed [1], [19], [27]. For other graph problems, see e.g., [28]. We also remark that the algorithms by Badanidiyuru and Vondrák [4] can be viewed as multi-pass streaming algorithms; using $O(\varepsilon^{-1}\log(\varepsilon^{-1}K))$-passes for a cardinality constraint, and $O(\varepsilon^{-2}\log^2(\varepsilon^{-1}n))$-passes for a $p$-system constraint. Our results deal with a knapsack constraint with fewer passes.

The maximum coverage problem is a special case of monotone submodular maximization under a cardinality constraint where the function is a set-covering function. For the special case, McGregor and Vu [29] gave a $(1-e^{-1}-\varepsilon)$-approximation algorithm in the multi-pass streaming setting. They use a sampling technique to estimate the value of $f(\mathrm{OPT})$ and then collect items based on thresholds using $O(\varepsilon^{-1})$ passes. Batani *et al.* [5] independently proposed a streaming algorithm with a sketching technique for the same problem.

**Acknowledgments** This work was supported by JST ERATO Grant Number JPMJER1201 and JSPS KAKENHI Grant Numbers JP18H05291 and JP17K00028, Japan.

## References

[1] Ahn, K. J. and Guha, S.: Linear programming in the semi-streaming model with application to the maximum matching problem, *Information and Computation*, Vol. 222, pp. 59 – 79 (2013). 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[2] Alon, N., Gamzu, I. and Tennenholtz, M.: Optimizing budget allocation among channels and influencers, *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pp. 381–388 (2012).

[3] Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A. and Krause, A.: Streaming submodular maximization: massive data summarization on the fly, *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 671–680 (2014).

[4] Badanidiyuru, A. and Vondrák, J.: Fast algorithms for maximizing submodular functions, *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1497–1514 (2013).

[5] Bateni, M., Esfandiari, H. and Mirrokni, V.: Almost Optimal Streaming Algorithms for Coverage Problems, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, New York, NY, USA, ACM, pp. 13–23 (online), DOI: 10.1145/3087556.3087585 (2017).

[6] Calinescu, G., Chekuri, C., Pál, M. and Vondrák, J.: Maximizing a Monotone Submodular Function Subject to a Matroid Constraint, *SIAM Journal on Computing*, Vol. 40, No. 6, pp. 1740–1766 (2011).

[7] Chakrabarti, A. and Kale, S.: Submodular maximization meets streaming: matchings, matroids, and more, *Mathematical Programming*, Vol. 154, No. 1-2, pp. 225–247 (2015).

[8] Chan, T.-H. H., Huang, Z., Jiang, S. H.-C., Kang, N. and Tang, Z. G.: Online Submodular Maximization with Free Disposal: Randomization Beats for Partition Matroids Online, *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1204–1223 (2017).

[9] Chan, T.-H. H., Jiang, S. H.-C., Tang, Z. G. and Wu, X.: Online Submodular Maximization Problem with Vector Packing Constraint, *Annual European Symposium on Algorithms (ESA)*, pp. 24:1–24:14 (2017).

[10] Chekuri, C., Gupta, S. and Quanrud, K.: Streaming Algorithms for Submodular Function Maximization, *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, Vol. 9134, pp. 318–330 (2015).

[11] Chekuri, C., Vondrák, J. and Zenklusen, R.: Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes, *SIAM Journal on Computing*, Vol. 43, No. 6, pp. 1831–1879 (2014).

[12] Ene, A. and Nguyen, H. L.: A Nearly-linear Time Algorithm for Submodular Maximization with a Knapsack Constraint (2017). `https://arxiv.org/abs/1709.09767`

[13] Feldman, M., Karbasi, A. and Kazemi, E.: Do Less, Get More: Streaming Submodular Maximization with Subsampling, *CoRR*, Vol. abs/1802.07098 (2018).

[14] Filmus, Y. and Ward, J.: A Tight Combinatorial Algorithm for Submodular Maximization Subject to a Matroid Constraint, *SIAM Journal on Computing*, Vol. 43, No. 2, pp. 514–542 (2014).

[15] Fisher, M. L., Nemhauser, G. L. and Wolsey, L. A.: An Analysis of Approximations for Maximizing Submodular Set Functions I, *Mathematical Programming*, pp. 265–294 (1978).

[16] Fisher, M. L., Nemhauser, G. L. and Wolsey, L. A.: An Analysis of Approximations for Maximizing Submodular Set Functions II, *Math-*

*ematical Programming Study*, Vol. 8, pp. 73–87 (1978).
[17] Huang, C.-C., Kakimura, N. and Yoshida, Y.: Streaming Algorithms for Maximizing Monotone Submodular Functions under a Knapsack Constraint, *The 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems(APPROX2017)* (2017).
[18] Huang, C.-C., Kakimura: Multi-Pass Streaming Algorithms for Monotone Submodular Function Maximization (2018). `https://arxiv.org/abs/1802.06212`
[19] Kale, S. and Tirodkar, S.: Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams, *The 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems(APPROX2017)* (2017).
[20] Kempe, D., Kleinberg, J. and Tardos, É.: Maximizing the spread of influence through a social network, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 137–146 (2003).
[21] Krause, A., Singh, A. P. and Guestrin, C.: Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies, *Journal of Machine Learning Research*, Vol. 9, pp. 235–284 (2008).
[22] Kulik, A., Shachnai, H. and Tamir, T.: Maximizing Submodular Set Functions Subject to Multiple Linear Constraints, *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 545–554 (2013).
[23] Lee, J.: *Maximum Entropy Sampling*, Encyclopedia of Environmetrics, Vol. 3, pp. 1229–1234, John Wiley & Sons, Ltd. (2006).
[24] Lee, J., Sviridenko, M. and Vondrák, J.: Submodular Maximization over Multiple Matroids via Generalized Exchange Properties., *Mathematics of Operations Research*, Vol. 35, No. 4, pp. 795–806 (2010).
[25] Lin, H. and Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions, *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 912–920 (2010).
[26] Lin, H. and Bilmes, J.: A class of submodular functions for document summarization, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pp. 510–520 (2011).
[27] McGregor, A.: Finding Graph Matchings in Data Streams, *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems (APPROX05)*, pp. 170–181 (2005).
[28] McGregor, A.: Graph Stream Algorithms: A Survey, *SIGMOD Rec.*, Vol. 43, No. 1, pp. 9–20 (2014).
[29] McGregor, A. and Vu, H. T.: Better Streaming Algorithms for the Maximum Coverage Problem, *International Conference on Database Theory (ICDT)* (2017).
[30] Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J. and Krause, A.: Lazier Than Lazy Greedy, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, AAAI Press, pp. 1812–1818 (2015).
[31] Mirzasoleiman, B., Jegelka, S. and Krause, A.: Streaming Non-monotone Submodular Maximization: Personalized Video Summarization on the Fly, *Proc. Conference on Artificial Intelligence (AAAI)* (2018).
[32] Soma, T., Kakimura, N., Inaba, K. and Kawarabayashi, K.: Optimal Budget Allocation: Theoretical Guarantee and Efficient Algorithm, *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 351–359 (2014).
[33] Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint, *Operations Research Letters*, Vol. 32, No. 1, pp. 41–43 (2004).
[34] Wolsey, L.: Maximising real-valued submodular functions: primal and dual heuristics for location problems, *Mathematics of Operations Research* (1982).
[35] Yoshida, Y.: Maximizing a Monotone Submodular Function with a Bounded Curvature under a Knapsack Constraint (2016). `https://arxiv.org/abs/1607.04527`
[36] Yu, Q., Xu, E. L. and Cui, S.: Streaming Algorithms for News and Scientific Literature Recommendation: Submodular Maximization with a $d$-Knapsack Constraint, *IEEE Global Conference on Signal and Information Processing* (2016).