Size-Constrained k-Submodular Maximization in Near-Linear Time (Supplementary Material)

Guanyu Nie[1] **Yanhui Zhu**[1] **Yididiya Y. Nadew**[1] **Samik Basu**[1] **A. Pavan**[1] **Christopher John Quinn**[1]

[1]Computer Science Deptartment, Iowa State University, Ames, IA, USA

A THRESHOLD GREEDY – INDIVIDUAL SIZE CONSTRAINTS

In this section, we will prove Theorem 5. We first recall the statement.

Theorem 5: Algorithm 2 runs in $O(kn\varepsilon^{-1}\log(B\varepsilon^{-1}))$ and guarantees a $(1/3-\varepsilon)$-approximation.

Proof. We first prove the run-time and then the approximation ratio.

Run-time: The **for** loop runs over all item-type pairs that could feasibly be added to S, thus taking $\mathcal{O}(nk)$ time each call. The number of times the outer **while** loop is called is equal to the smallest integer t' such that $(1-\varepsilon)^{t'}d \leq \frac{(1-\varepsilon)\varepsilon d}{3B}$. Let t denote the value where equality holds, so $t' = \lceil t \rceil$. Rearranging, t satisfies

$$\begin{aligned} & t\log(1-\varepsilon) = \log(1-\varepsilon) - \log(3B\varepsilon^{-1}) \\ \iff \quad & t = 1 - \frac{\log(3B\varepsilon^{-1})}{\log(1-\varepsilon)} && (\log(1-\varepsilon) < 0) \\ & \leq 1 + \frac{\log(3B\varepsilon^{-1})}{\varepsilon} && (\text{using the fact that } \log(1-x) < -x \text{ for } x < 1) \\ \iff \quad & t' = \lceil t \rceil \leq 2 + \frac{\log(3B\varepsilon^{-1})}{\varepsilon}. \end{aligned}$$

Thus, with $\mathcal{O}(\varepsilon^{-1}\log(B\varepsilon^{-1}))$ calls of the outer **while** loop, the total run time is $\mathcal{O}(nk\varepsilon^{-1}\log(B\varepsilon^{-1}))$.

Approximation Guarantee: The analysis for Algorithm 2's performance with individual size (IS) constraints will, broadly speaking, resemble the analysis of Algorithm 1's performance for total size (TS) constraints. Like in the proof for Theorem 4, we will construct a sequence of feasible solutions and relate objective value differences between successive pairs of solutions to the marginal gains achieved with each item-type pair added in Algorithm 2. However, the construction will require more care as the swapping pairs must respect the constraints of each type, not simply the cardinality of B. Note that while we will reuse "B" as $B \leftarrow \sum_{i=1}^{k} B_i$.

We will first consider the case that Algorithm 2 outputs a maximal solution, one that for each type $i \in [k]$ has B_i elements assigned that type. We will then consider the general case. Also, we consider that OPT is also maximal, $|U_i(\mathrm{OPT})| = B_i$ for all $i \in [k]$. This is without loss of generality as the monotonicity of f implies that if there is an optimal solution with less than maximal cardinality, we can add in elements to it without a decrease in value.

Case 1: The final solution S° satisfies $|U_i(S^\circ)| = B_i$ for all $i \in [k]$.

We reuse the notation (e_j, i_j) for the jth item-type pair that was added to $S^\circ = \{(e_1, i_1), \ldots, (e_B, i_B)\}$ by Algorithm 2. Similar to the total size case, we will again construct several sequences combining S and OPT to show inequalities resulting in the stated approximation bound. We next let S_j denote S after j elements were added, so $S_j := \{(e_1, i_1), \ldots, (e_j, i_j)\}$

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

and we set $S_0 := \emptyset$ as the initial empty set. Thus by construction

$$f(S_{j+1}) - f(S_j) = f((e_{j+1}, i_{j+1})|S_j).$$

We next index the item-type pairs in the optimal solution $\text{OPT} = \{(e'_1, i'_1), \ldots, (e'_B, i'_B)\}$. If the item e_j in the jth pair $(e_j, i_j) \in S^\circ$ of the output is also in a pair (e_j, i') in OPT, the latter pair should have the same index. For other pairs in OPT, the indexing is arbitrary. With this alignment of indices of pairs in S° and OPT that share a common item, we construct a sequence of cardinality B sets $O_0, O_1, \ldots, O_B$. We will not be able to simply swap pairs in a single position, as we did in the proof for Theorem 4, but will need to swap additional pairs to maintain feasibility of with respect to all type constraints $\{B_i\}_{i=1}^k$. Like in the TS case, we will want the beginning and end of the sequence of feasible solutions to match the optimal and output solutions,

$$\begin{aligned} O_0 := \text{OPT} &= \{(e'_1, i'_1), (e'_2, i'_2), \ldots, (e'_{B-1}, i'_{B-1}), (e'_B, i'_B)\} \\ O_B := S^\circ &= \{(e_1, i_1), (e_2, i_2), \ldots, (e_{B-1}, i_{B-1}), (e_B, i_B)\} \end{aligned}$$

We will construct the sequence beginning with $O_0 = \text{OPT}$. If we construct O_1 by replacing the first pair in O_0 with the first pair in S°,

$$\{(e_1, i_1), (e'_2, i'_2), \ldots, (e'_{B-1}, i'_{B-1}), (e'_B, i'_B)\},$$

if the types in those two pairs were different, $i_1 \neq i'_1$, then since OPT already had B_{i_1} elements of type i_1, it would now have $B_{i_1} + 1$ elements of type i_1, thus violating the constraint. To ensure that O_1 is feasible, we first swap types for two pairs in OPT to make sure the first pair has type i_1, and then swap elements in the first pair to match S_1. Let ℓ denote any index of any pair in OPT with type i_1 (even $\ell = 1$ if we are in the safe case that the types already matched $i_1 = i'_1$; the following inequalities will still hold). We introduce $O_{0+1/2}$ that swaps types, before swapping elements to construct O_1,

$$O_0 := \text{OPT} = \{(e'_1, i'_1), (e'_2, i'_2), \ldots, (e'_\ell, i'_\ell = i_1), \ldots, (e'_B, i'_B)\}$$

$$O_{0+1/2} := \{(e'_1, i'_\ell = i_1), (e'_2, i'_2), \ldots, (e'_\ell, i'_1), \ldots, (e'_B, i'_B)\} \qquad \text{(swap types of pairs 1 and } \ell\text{)}$$

$$O_1 := \{(e_1, i_1), (e'_2, i'_2), \ldots, (e'_\ell, i'_1), \ldots, (e'_B, i'_B)\}. \qquad \text{(swap element in pair 1)}$$

To construct O_2, we will do a similar set of swaps. Let $h \in \{2, \ldots, B\}$ denote the index of a pair in O_1 with type i_2 (the same type as the second element (e_2, i_2) added greedily to S° has).

$$O_1 := \{(e_1, i_1), (e'_2, i'_2), \ldots, (e'_\ell, i'_1), \ldots, (e'_h, i'_h = i_2), \ldots, (e'_B, i'_B)\}$$

$$\begin{aligned} O_{1+1/2} &:= \{(e_1, i_1), (e'_2, i'_h = i_2), \ldots, (e'_\ell, i'_1), \ldots, (e'_h, i'_2), \ldots, (e'_B, i'_B)\} && \text{(swap types of pairs 2 and } h\text{)} \\ O_2 &:= \{(e_1, i_1), (e_2, i_2), (e'_2, i'_2), \ldots, (e'_\ell, i'_1), \ldots, (e'_h, i'_2), \ldots, (e'_B, i'_B)\}. && \text{(swap element in pair 2)} \end{aligned}$$

We continue in this fashion, while constructing O_{j+1} for $j \in \{1, \ldots, B-1\}$ looking for a pair in O_j with type i_{j+1} (i.e. the same type as in the $(j+1)$st pair added to S°) among indices $r \in \{j+1, \ldots, B\}$. Since in swapping we match types to align with the types in S_{j+1}, and S° was feasible, there will always be such an index r. As noted, if the types are already aligned in position $j+1$ in O_j, and r is chosen as $j+1$, then $O_j = O_{j+1/2}$ and the following work will still hold (some inequalities will be loose).

Note that by construction, for $j \in \{0, \ldots, B-1\}$ we have $S_j \subseteq O_j \cap O_{j+1/2} \cap O_{j+1}$.

We now consider the difference $f(O_j) - f(O_{j+1})$. This is not a marginal gain since neither set contains the other. However, since the sets differ in the $(j+1)$st index and possibly one more index $r \in \{j+1, \ldots, B\}$, we will be able to upper bound the difference in terms of the marginal gain $f(S_{j+1}) - f(S_j)$ achieved by Algorithm 2 in adding the $(j+1)$st element (e_{j+1}, i_{j+1}).

Let $r \in \{j+1, \ldots, B\}$ denote the index of the pair in O_j that we swapped types with the $(j+1)$st pair.

As types may have been changed in positions $j+1$ and/or r multiple times due to previous swaps while constructing $\{O_1, \ldots, O_j\}$, let $(e'_{j+1}, \tilde{i}_{j+1})$ and $(e'_r, \tilde{i}_r)$ denote the pairs in those positions in O_j. Those pairs have the same items as in

$O_0 = \text{OPT}$ but the types may differ due to previous swaps. In general, we will have

$$\begin{aligned}
f(O_j) - f(O_{j+1}) &= \big(f(O_j \cap O_{j+1}) + f(\{(e'_{j+1}, \tilde{i}_{j+1}), (e'_r, \tilde{i}_r)\}|O_j \cap O_{j+1})\big) \\
&\quad - \big(f(O_j \cap O_{j+1}) - f(\{(e_{j+1}, i_r), (e'_r, \tilde{i}_{j+1})\}|O_j \cap O_{j+1})\big) && \text{(def. of marginal gains)} \\
&= f(\{(e'_{j+1}, \tilde{i}_{j+1}), (e'_r, \tilde{i}_r)\}|O_j \cap O_{j+1})\big) - f(\{(e_{j+1}, i_r), (e'_r, \tilde{i}_{j+1})\}|O_j \cap O_{j+1})\big) \\
& && \text{(cancel common terms)} \\
&\leq f(\{(e'_{j+1}, \tilde{i}_{j+1}), (e'_r, \tilde{i}_r)\}|O_j \cap O_{j+1})\big) && \text{(by monotonicity marginal gains are non-negative)} \\
&\leq f(\{(e'_{j+1}, \tilde{i}_{j+1})\}|O_j \cap O_{j+1})\big) + f(\{(e'_r, \tilde{i}_r)\}|O_j \cap O_{j+1})\big). && \text{(by submodularity)} \\
&\leq f((e'_{j+1}, \tilde{i}_{j+1})|S_j) + f(\{(e'_r, \tilde{i}_r)\}|S_j)\big), && (1)
\end{aligned}$$

Where the last inequality follows by submodularity and $S_j \subseteq O_j \cap O_{j+1}$. We next determine that both $(e'_{j+1}, \tilde{i}_{j+1})$ and $(e'_r, \tilde{i}_r)$ were feasible pairs to add to S_j in Algorithm 2 when (e_{j+1}, i_{j+1}) was selected. Considering each of those terms, we note that by construction, we aligned indices of pairs in OPT that included items that were in pairs in S°. Thus, we have that $e'_{j+1} \notin S_j$ and $e'_r \notin S_j$, meaning that both elements were still available for Algorithm 2 to pick. Additionally, by construction (since we swap to align types between $j+1$st pairs) neither type $\tilde{i}_{j+1}$ nor $\tilde{i}_r = i_{j+1}$ had been exhausted when Algorithm 2 selected (e_{j+1}, i_{j+1}) to add to S_j. Thus, since Algorithm 2 picked (e_{j+1}, i_{j+1}) instead of either $(e'_{j+1}, \tilde{i}_{j+1})$ or $(e'_r, \tilde{i}_r)$, the corresponding marginal gains could not have been much larger than that of (e_{j+1}, i_{j+1}). Namely, with τ_{j+1} denoting the threshold when (e_{j+1}, i_{j+1}) was selected by Algorithm 2, then

$$f((e_{j+1}, i_{j+1})|S_j) \geq \tau_{j+1}$$

and since the other two pairs were not selected in the previous round when the threshold was $(1-\varepsilon)^{-1}\tau_{j+1}$ (or, if $\tau_j = d$, the maximum marginal gain, then they are at most equal and the following still holds),

$$f((e'_{j+1}, \tilde{i}_{j+1})|S_j) \leq (1-\varepsilon)^{-1} f((e_{j+1}, i_{j+1})|S_j)$$

and

$$f((e'_r, \tilde{i}_r)|S_j) \leq (1-\varepsilon)^{-1} f((e_{j+1}, i_{j+1})|S_j)$$

This allows us to continue (1):

$$\begin{aligned}
f(O_j) - f(O_{j+1}) &\leq f((e'_{j+1}, \tilde{i}_{j+1})|S_j) + f(\{(e'_r, \tilde{i}_r)\}|S_j)\big) && \text{(by (1))} \\
&\leq 2(1-\varepsilon)^{-1} f((e_{j+1}, i_{j+1})|S_j). && (2)
\end{aligned}$$

Using this relation, we can now lower bound $f(S^\circ)$.

$$\begin{aligned}
f(\text{OPT}) - f(S^\circ) &= \sum_{j=0}^{B-1} (f(O_j) - f(O_{j+1}) && \text{(telescoping sum)} \\
&\leq \sum_{j=0}^{B-1} \frac{2}{1-\varepsilon} (f(S_{j+1}) - f(S_j)) && \text{(by (2))} \\
&= \frac{2}{1-\varepsilon} (f(S^\circ) - f(\emptyset)) \\
&\leq \frac{2}{1-\varepsilon} f(S^\circ)
\end{aligned}$$

which for $\varepsilon < 1$ implies

$$\begin{aligned}
f(S^\circ) &\geq \frac{1-\varepsilon}{3-\varepsilon} f(\text{OPT}) \\
&\geq (\frac{1}{3} - \varepsilon) f(\text{OPT}). && (3)
\end{aligned}$$

Case 2: The final solution S° satisfies $|U_i(S^\circ)| < B_i$ for some $i \in [k]$. Let $\ell_i = |U_i(S)| < B_i$ denote the number of items with type i added. Let $\tilde{S}$ denote a set of cardinality B that Algorithm 1 would have selected if Algorithm 1 terminated only when either (a) B_i pairs had been selected for any type i or (b) the marginal gains on all remaining elements evaluated as zero. Without loss of generality, we only consider (a), as (b) would imply $f(\tilde{S}) = f(\mathrm{OPT})$ and subsequently the same bounds as we will show for (a). Thus, by construction $S \subset \tilde{S}$ and $\tilde{S}$ has $\sum_{i\in[k]}(B_i - \ell_i)$ extra elements.

First, since $\tilde{S}$ has B elements selected according to decreasing thresholds, the result (3) from **Case 1** holds for $\tilde{S}$, that for $\varepsilon < 1$,

$$f(\tilde{S}) \geq \frac{1-\varepsilon}{3-\varepsilon} f(\mathrm{OPT}). \tag{4}$$

Second, since S only accumulated $\sum_{i\in[k]} \ell_i$ elements before the terminal threshold bound of $\frac{(1-\varepsilon)\varepsilon d}{3B}$ was reached, then the marginal gains of the remaining $\sum_{i\in[k]}(B_i - \ell_i)$ elements in $\tilde{S}$ can be bounded, with the largest possible value of the threshold τ in the last execution of the **while** loop being

$$(1-\varepsilon)^{-1}\frac{(1-\varepsilon)\varepsilon d}{3B} = \frac{\varepsilon d}{3B},$$

leads to

$$\begin{aligned}
f(\tilde{S}) - f(S^\circ) &\leq \sum_{(e,i)\in\tilde{S}\setminus S^\circ} f((e,i)|S^\circ) && \text{(using Lemma 1 in main paper)} \\
&\leq \sum_{(e,i)\in\tilde{S}\setminus S^\circ} \frac{\varepsilon d}{3B} \\
&= \sum_{i\in[k]} (B_i - \ell_i)\frac{\varepsilon d}{3B} \\
&\leq \frac{\varepsilon d}{3} && (\textstyle\sum_{i\in[k]}(B_i - \ell_i) \leq B) \\
\iff \qquad f(S^\circ) &\geq f(\tilde{S}) - \frac{\varepsilon d}{3}. && (5)
\end{aligned}$$

We note that since by construction $S^\circ \subset \tilde{S}$, each of the item-index pairs in $\tilde{S}\setminus S^\circ$ must have items not in $U(S^\circ)$, the marginal gains in the formulas above are well-defined.

Combining (4) and (5),

$$\begin{aligned}
f(S^\circ) &\geq f(\tilde{S}) - \frac{\varepsilon d}{3} && \text{(by (5))} \\
&\geq \frac{1-\varepsilon}{3-\varepsilon} f(\mathrm{OPT}) - \frac{\varepsilon d}{3} && \text{(by (4))} \\
&\geq \frac{1-\varepsilon}{3-\varepsilon} f(\mathrm{OPT}) - \frac{\varepsilon f(\mathrm{OPT})}{3} && \text{(by submodularity and choice of } d) \\
&\geq (\frac{1}{3} - \varepsilon) f(\mathrm{OPT}).
\end{aligned}$$

□

B OTHER RELATED WORKS NOT CONSIDERED AS BASELINE

There are some related works mentioned in Table 1 but not considered as baselines. We provide detailed reasons and discussions here:

- [Qian et al., 2017]: The authors propose an evolutionary algorithm. They only proposed and analyzed the algorithm for total size constraints, not for individual size constraints. We note that the algorithm runs continuously; the run time

shown in our Table 1 is only an expectation of time for the algorithm to obtain the desired approximation guarantee of $1/2$. Their code is publicly available, though was implemented for sensor placement experiments using a small portion of the data. We modified it to run for the whole data set, but those experiments are slow compared to other methods. One fundamental reason is that the evolutionary algorithm cannot incorporate lazy evaluation. This is significant as in the example of sensor placement with $k = 3$, total budget of 36, the greedy algorithm without lazy evaluation would require 3,888 function evaluations (using lazy evaluation, only 1627 were required as shown in Figure 1c). Also, the mutation process takes $\mathcal{O}(n)$ time even if the mutation is not accepted. While all the algorithms considered in the paper run within minutes, the evolutionary algorithm did not finish a case for total budget $B = 5$ within 2 hours. As the main purpose of the paper is to improve the time complexity of existing algorithms, the run time of the proposed evolution algorithm is much worse than the greedy algorithm (even without lazy evaluation), so we do not include it.

- [Ene and Nguyen, 2022]: This paper considers the streaming setting, where base elements arrive one at a time in an arbitrary (adversarial) order. We did not think it would be fair to compare it to offline methods like our threshold greedy method, the stochastic greedy, or greedy methods.
- [Matsuoka and Ohsaka, 2021]: This paper does not propose a new algorithm. The authors analyze the greedy algorithm (which we include in experiments). The authors prove the greedy algorithm achieves a better approximation ratio for the sub-class of k-submodular functions with bounded curvature.

References

Alina Ene and Huy Nguyen. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *International Conference on Machine Learning*, pages 5944–5967. PMLR, 2022.

Tatsuya Matsuoka and Naoto Ohsaka. Maximization of monotone k-submodular functions with bounded curvature and non-k-submodular functions. In *Asian Conference on Machine Learning*, pages 1707–1722. PMLR, 2021.

Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22 (4):595–608, 2017.