Interlaced Greedy Algorithm for Maximization of Submodular Functions in Nearly Linear Time

Alan Kuhnle
Florida State University

**Submodular Functions**

**Definition** A function \( f : 2^U \rightarrow \mathbb{R}_{\geq 0} \) defined on subsets of a ground set \( U \) of size \( n \) is **submodular** if it possesses the following property:

- For all \( A \subseteq B \subseteq U \) and \( x \notin B \),
  \[
  f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)
  \]

**Maximization Subject To Cardinality Constraint**

**Definition** Given \( k \), submodular function \( f : 2^U \rightarrow \mathbb{R}_{\geq 0} \), find

\[
\text{argmax}_{|S| \leq k} f(S). \quad \text{(MCC)}
\]

**Example** Social Network Monitoring. Suppose we have a social network \( G = (V, E) \), where each edge \((u, v)\) has weight \( w(u, v)\) according to how much content (e.g. status updates, news stories, videos) is shared from \( u \) to \( v \). Let \( S \subseteq V \), and define

\[
f(S) = \sum_{u \in S, v \notin S} w(u, v).
\]

Intuitively, \( f(S) \) encodes how much content flows from \( S \) to \( V \setminus S \).

Then MCC asks for the set of size at most \( k \) through which the most content passes.

**Other Applications**
- Video summarization
- Movie recommendation
- Revenue maximization

**Interlaced Greedy Approach**

Two standard greedy procedures are interlaced. That is, elements are greedily selected into one of two sets in alternating fashion.

- Maintain two disjoint sets \( A \) and \( B \), initially empty.
- For \( k \) iterations, the best element \((\text{argmax}_{x \in A} f_x(A))\) is selected to add to \( A \), then the best element \((\text{argmax}_{x \notin A} f_x(B))\) is added to \( B \).
- For technical reasons, perform an analogous procedure starting from \( A = B = \{a_0\} \).

**Stealing Heuristic**

- From interlacing, unequivocally good elements may be divided between \( A \) and \( B \).
- Let \( C \) be the set returned and \( U \) be the set of elements selected by any greedy procedure but not included in \( C \). Sort \( C \) by increasing value of \( f(C) - f(C \setminus \{c\}) \) and sort \( U \setminus C \) by decreasing value of \( f(C \cup \{x\}) - f(C) \). In one pass through \( C \), test if swapping element \( c_i \in C \) with \( x_i \in U \setminus C \) yields an improvement in \( f(C) \).
- Time complexity: \( O(k \log k) \).

**Speeding it up**

- Standard greedy \( \rightarrow \) thresholded greedy procedures
- Thresholded greedy procedure: in one pass through \( U \), add elements whose marginal gain exceed a threshold \( \tau \). Repeat for \( O(\log k) \) suitably chosen thresholds.

**Ratio:** \( 1/4 - \delta \)

**Time complexity:** \( O\left(\frac{n}{\delta} \log \left(\frac{k}{\delta}\right)\right)\)

---

**Experimental Evaluation**

- (a) BA, Cut Value
- (b) BA, Function Queries

Figure 1: Objective value and runtime for cardinality-constrained maxcut.

- (a) ER instance, \( n = 1000 \)
- (b) BA instance, \( n = 10000 \)

Figure 2: Effect of stealing procedure on solution quality of FIG.

*Contact: akuhle@fsu.edu, https://www.alankuhnle.com/*