Involve solving the Problem 1 with submodular function $f$:

$$\max_{|S|<t} f(S).$$

### Challenges in Big Data:
- Greedy algorithm is centralized and sequential.
- Even lazy greedy algorithm (LAZYGREED) does not scale well.

### Contributions:
- Introduce a multi-stage framework (MULTIGREED) to speed up LAZYGREED in 3 ways:
  1. reduce the number of function evaluations (APPROXGREED),
  2. reduce the ground set size (Pruning),
  3. decrease the complexity of function evaluations (Surrogate functions),
- Complementary to existing distributed algorithms.
- Generalizable to submodular knapsack problem and submodular set cover problem.

### Multi-stage Algorithmic Framework

**Approximate Greedy (APPROXGREED($f, \ell, \beta_{i=1}^{\ell}$)):**

- Not finding the item that attains the maximum gain in each iteration.
- Only looking for an item whose gain is at least $\beta_i \leq 1$ of maximum gain.

**Lemma**

**Pruning:** Let $\{u_i\}_{i=1}^{\ell}$ be such that $f(u_i|V \setminus u_i) \geq \cdots \geq f(u_0|V \setminus u_0)$, then LAZYGREED on the reduced set $V = \{j \in V|f(j) \geq f(u_i|V \setminus u_i)\}$ is equivalent to that applied on the ground set $V$.

**MULTIGREED** may optionally start with the pruning step.
- Can be implemented in parallel.

**Algorithm 1 Multi-stage Framework (MULTIGREED)**

1. Input: a submodular function $f$, cardinality constraint $t$, number of stages $\ell$, proxy functions $(f_j)_{j=1}^{\ell}$, size constraints $(\beta_j)_{j=1}^{\ell}$, and $(\beta_{i=1}^{\ell})$.
2. Initialize $C' = \emptyset$, $L = 0$; 3. for $j = 1$ to $\ell$ do
   4. Define $F_j(S) \equiv f_j(S|C)$ for all $S \subseteq V$
5. $S \in$ APPROXGREED($f_j, \beta_j, (\beta_{i=1}^{\ell})$)
6. $L = L + \ell$, $C' = C' + S$
end for
8. Output $C$

### Theoretical Analysis

**Greedy ratio:** Harmonic mean of the individual greedy ratios $\{a_i\}_{i=1}^{\ell}$.

$$\alpha = \frac{\ell}{\sum_{i=1}^{\ell} \frac{1}{a_i}}$$

where individual greedy ratio is defined as:

$$a_i = \max_{S \subseteq V} f(S|S_i, S_{\ell}) \in [1, +\infty]$$

- $a_i$: ratio of greedy gain to the gain of $S_i$ (chosen by MULTIGREED).

#### Curvature:

$$\kappa_{\ell}(S) = 1 - \min_{V \setminus V} \frac{\nu \ell}{f(S \setminus S_{\ell})} \in [0, 1]$$

**Theorem**

An instance of MULTIGREED with greedy ratio $\alpha$ is guaranteed to obtain a set $S$ s.t.

$$\frac{f(S)}{f(S_{OPT})} \geq \frac{1}{\alpha} - \left(1 - \frac{1}{\alpha}\right)^{\beta_{j=1}^{\ell}} \geq \frac{1}{\alpha} - (1 - \epsilon^2) \geq (1 - \epsilon)$$

Conversely, for any $\alpha > \kappa_{\ell}$, there exists an $f$ with curvature $\kappa_{\ell}$, on which instance of MULTIGREED with greedy ratio $\alpha$ achieves an approximation factor $\frac{1}{\alpha} - (1 - \epsilon^2)$.

**Corollary:** APPROXGREED($f, \ell, (\beta_{i=1}^{\ell})$) is guaranteed to achieve a factor of $(1 - \epsilon)$, where $\epsilon = 1/\sum_{i=1}^{\ell} \beta_i$.

**Greedy ratio** also work in other scenarios such as submodular knapsack and submodular set cover problems.

**Goal:** design MULTIGREED such that its greedy ratio is close to 1.

### Surrogate Functions

**Uniform Submodular Mixtures:**

$$f(S) = \frac{1}{|T|} \sum_{f \in T} f(S)$$

where $|T| \geq 1$, and $f$ is monotone submodular $\forall f \in T$.

**Lemma**

Using $f_{\text{sub}}$ as a surrogate, it holds that $1 - \delta$, where $\delta = (1 - 5\epsilon) \frac{\ell}{\ell + 1}$.

**Modular Surrogate:**

$$f_{\text{mod}}(S) = \sum_{f \in T} f(S)$$

where $T' \subseteq T$ is generated by sampling from $T$ with probability $p$.

**Lemma**

Using $f_{\text{mod}}$ as a surrogate, it holds that $1 - \alpha_i$, where $\alpha_i = \frac{1}{\ell + 1}$.

**Graph-based Functions:** Defined via an underlying weighted complete graph.

**k-NNG Surrogate:** $f_{\text{k-NNG}}$ defined on a k-NNG for a graph-based function $f$.

### Experiments

#### Simulations:
- Vary $k$, $p$, and $\ell$ for $f_{\text{fac}}, f_{\text{sat}}$, and $f_{\text{rea}}$ respectively.

#### Speech Data Subset Selection:
- A real-world and large-scale ($n = 1, 322, 018$) problem.
- Solve Problem 1 with $f_{\text{fac}}$.
- Run MULTIGREED ($\ell = 1$) with $f_{\text{fac}}$.
- Yield $\geq 1000$ times speedup over LAZYGREED.

#### Table:

<table>
<thead>
<tr>
<th>Facility Function</th>
<th>Random</th>
<th>Histogram</th>
<th>Entropy</th>
<th>Multi-stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{\text{fac}}$</td>
<td>38.2</td>
<td>35.1</td>
<td>34.4</td>
<td>37.3</td>
</tr>
<tr>
<td>$f_{\text{sat}}$</td>
<td>37.6</td>
<td>34.2</td>
<td>31.0</td>
<td>34.1</td>
</tr>
<tr>
<td>$f_{\text{rea}}$</td>
<td>32.7</td>
<td>34.1</td>
<td>32.7</td>
<td>37.3</td>
</tr>
</tbody>
</table>

| Averaged Random   | 38.2   | 35.1      | 34.4    | 37.3        |
| 5% 10% 20% 50% 80%| 38.2   | 35.1      | 34.4    | 37.3        |

**Acknowledgements:** This work was partially supported by IRAP under agreement number FA8650-12-2-7263, the NSF under Grant No. IIS-1162606, and by a Google, a Microsoft, and an Intel research award.